Dynamic Access Control Framework For Internet of Things

by

Ashraf Alkhresheh

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
December 2019

Abstract

In the near future, IoT ecosystems will enable billions of smart things to interconnect and communicate information about themselves and their physical environments. The high density of smart things in these environments allows for fine-grained data acquisition, enabling the development of advanced services and new kinds of applications ranging from wearable devices to air conditioners to fully automated cars. However, the dense and pervasive collection, processing and dissemination of data can unleash sensitive information about individuals, raising non-trivial security and privacy concerns. One solution for IoT security and privacy is to restrict access to sensitive data using access control and authorization techniques. Although many basic principles of standard access control models continue to apply, the high dynamic nature of IoT environments, resources limitation of IoT devices and vulnerability to physical and virtual attacks present unique challenges that render existing access control schemes unfit for IoT. This research introduces a holistic and dynamic access control framework for IoT environments. The framework consists of three components: an automatic and context-aware policy specification method, continuous policy enforcement mechanism and an adaptive policy adjustment technique. In response to access requests, the automatic policy specification component dynamically generates access control rules that grant access permissions based on predefined primitive facts. The primitive facts describe the attributes of the IoT devices registered to the system and the operational contexts under which these devices can interact. The continuous policy enforcement mechanism constantly monitors the compliance of the operational context while resource is in use, and re-evaluates ongoing access sessions in response to changes in operational contexts and/or access policies. The adaptive policy adjustment component assesses the access behaviour of the IoT devices, adjusts the access control policies based on device behavioral patterns and recommends policy adjustments to the policy administrator for final approval. Experimental results show that the proposed framework provides higher adaptability to the dynamic security and privacy requirements of IoT deployments as well as better flexibility in access control policy management.

Co-Authorship

1. A. Alkhresheh, K. Elgazzar and H. S. Hassanein,”DACIoT: Dynamic Access Control Framework for IoT deployments,” in *IEEE Internet of Things Journal.* (Submitted)

2. A. Alkhresheh, K. Elgazzar and H. S. Hassanein,”Access control models in IoT: A Survey of dynamic authorization perspective,” *ACM Computing Surveys.* (Submitted)

3. A. Alkhresheh, K. Elgazzar and H. S. Hassanein,”Adaptive Access Control Policies for IoT deployments,” *2020 17th USENIX Symposium on Networked Systems Design and Implementation.* (Submitted)

4. A. Alkhresheh, K. Elgazzar and H. S. Hassanein, ”CAPE: Continuous Access Policy Enforcement for IoT Deployments,” *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 2019*, pp. 1576-1581.

5. A. Alkhresheh, K. Elgazzar and H. S. Hassanein, ”Context-aware Automatic Access Policy Specification for IoT Environments,” *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, Limassol, 2018, pp. 793-799.

Acknowledgments

First and foremost, I thank God the most gracious and most merciful for blessing me with guidance, strength, and patience to complete this work.

I would like to express my sincere gratitude towards my family for the encouragement which helped me in completion of my Ph.D degree. The endless forms of support I received from my father Mohammad and my mother Houria since my childhood are unimaginable, immeasurable, and incomprehensible. I shall always be grateful and thankful for what you have done and sacrificed. I would also like to thank my beloved wife Safaa, my life-long partner and soul-mate her love, encouragement, and dedication light the path throughout my Ph.D, and my lovable daughters Raneem, Yara and Adeem who served as my inspiration to pursue this undertaking.

I would like to express my utmost gratitude to my supervisors Prof. Hossam Hassanein and Prof. Khalid Elgazzar for their selfless support of my Ph.D study and related research. Your guidance, patience, advice and feedback led to the successful completion of this work. It has been a great pleasure working with you, I owe you a lot. I have also been fortunate to have a great thesis committee: Prof. Mohammad Zulkernine, Prof. Abd-Elhamid Taha. Thank you for your constructive feedback and support throughout the past years.

I also take this opportunity to record my sincere thanks to TRL members for providing me with a stimulating and fun work environment. Special thanks go to Yaser Al Mtawa, Mohannad Alswailim, Abdellah Ibrahim, Mohammad Adel, Mohannad Alharthi, Amir Ibrahim, Galal Hassan and Ahmad Nagib. To the dearest Basia Palmer, thank you for the overwhelming love and care you surrounded me with. I could not have made it through this challenging experience without the help and support I received from you.

Special thanks to Tafila Technical University in Jordan for their financial support to go through this endeavor. I look forward towards returning part of the favor to this distinguished establishment.

Last, but definitely not least, I would like to thank my lifelong friend Abdullah Al Ma'aitah. You have been such a great, supportive friend for over 15 years. May we maintain this friendship, and may I return your constant favors!

Statement of Originality

I hereby certify that this Ph.D. thesis is original and that all ideas and inventions attributed to others have been properly referenced.

Ashraf Alkhresheh

December 23, 2019

Contents

List of Tables

List of Figures

List of Acronyms

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
ABAC	Attribute Based Access Control
ABE	Attribute Based Encryption
AC	Access Control
ACC	Access Control Contract
ACL	Access Control List
ACM	Access Control Matrix
AdRBAC	Adaptive Risk-Based Access Control
APA	Adaptive Policy adjustment
APS	Automatic Policy Specification
AR	Access Request
AS	Authorization Server
AUC	Area Under the Curve
BLE	Bluetooth Low Energy
CAAC	Context Aware Access Control
CapBAC	Capability Based Access Control
CCAAC	Capability-based Context Aware Access Control
CGC	Common Guard Context

CH	Context Handler
CoAP	Constrained Application Protocol
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
CPE	Continuous Policy Enforcement
CPU	Central Processing Unit
DAC	Discretionary Access Control
DACIoT	Dynamic Access Control Framework for the Internet of Things
DCapBAC	Distributed Capability Based Access Control
DRM	Digital Rights Management
ECC	Elliptic Curve Cryptography
ECCDH	Elliptical Curve Cryptography-Diffie Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FTC	Federal Trade Commission
GC	Guard Context
GPS	Global Positioning System
GRBAC	Generalized Role-Based Access Control
HVAC	Heating, Ventilation, and Air Conditioning
HIPAA	Health Insurance Portability and Accountability Act
HTTPS	Hypertext Transfer Protocol Secure
IACAC	Identity Authentication and Capability-based Access Control
ICAP	Identity-based Capability System
IoT	Internet of Things
IPv6	Internet Protocol version 6
JC	Judge Contract

JSON	JavaScript Object Notation
KP-ABE	Key-Policy Attribute-Based Encryption
LSTM	Long Short-Term Memory
MAC	Mandatory Access Control
MLS	Multilevel Security Model
MQTT	Message Queuing Telemetry Transport
NFC	Near-Field Communication
NHTSA	National Highway Traffic Safety Administration
OASIS	Organization for the Advancement of Structured Information Standards
OC	Operational Context
OCTAVE	Operationally Critical Threat, Asset and Vulnerability Evaluation
OrBAC	Organizational Based Access Control
OST-R	The Office of the Assistant Secretary for Research and Technology
OWL	Web Ontology Language
PAA	Policy Adjustment Accuracy
PAP	Policy Administration Point
PD	Policy Database
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PF	Primitive Facts
PIP	Policy Information Point
RBAC	Role Based Access Control
RC	Register Contract
RF	Random Forest

RFID	Radio Frequency Identification
RNNs	Recurrent Neural Networks
ROC	Receiver Operating Characteristic
SAML	Security Assertion Markup Language
SM	Session Monitor
SNS	Social Network Services
SR	Session Registry
SWRL	Semantic Web Rule Language
UCON	Usage Access Control
UMA	User Managed Access
USDOT	United States Department of Transportation
W3C	World Wide Web Consortium
WoT	Web of Things
XACML	eXtensible Access Control Markup Language

Chapter 1

Introduction

Recent years have witnessed the emergence of the Internet of Things (IoT), a new computing paradigm that connects people with their surrounding environment to enable smart interactions. The traces of the IoT concept go back to the early work done by Kevin Ashton in 1999, which received enormous attention from both academia and industry all over the world [1]. Although many definitions have been proposed for the IoT concept [2], there is no single, globally accepted, definition for the term. The general idea behind IoT is to extend everyday objects not commonly considered computers, with the computing capability to identify themselves, generate, and communicate information about their physical environment [3]. Typically, these objects are physical and virtual sensors, Radio Frequency Identification (RFID) tags and smartphones [4].A mixture of enabling software and hardware technologies has contributed to the success of the IoT vision. These technologies can be grouped into two categories, (1) IoT functional supporting technologies such as RFID, GPS, cellular technologies, Wi-Fi, microcomputers and microprocessors, which add intelligence to the connected objects enabling them to acquire, communicate, and process contextual information and (2) IoT non-functional supporting technologies such as cryptography, authorization, and access control technologies, which improve the security and privacy of IoT environments by securing the data stored, processed and exchanged between IoT devices and data consumers, hence increasing the adoption of the IoT and utilizing the technology to its fullest potential.

The increasing number and density of connected IoT objects [5] coupled with the developments in the integrated technologies have widened the range of potential IoT applications over several domains. In the following, we provide a brief description of some examples of such IoT applications.

Logistic applications: With RFID, Near-field communication (NFC) and Global Positioning System (GPS) technologies, it is now possible to monitor, in real time, all supply chain stages including purchasing, production, inventory, distribution, routing, location, and marketing. IoT technology can make everything in the supply chain become connected, which allows for a vast amount of data to be analyzed and turned into facts that support decision-making and improve performance at each stage in the supply chain. For example, using IoT collected data can help in making more accurate demand forecasts, improve visibility at each step of the production process, prevent stock out and inventory shrinkage, improve collaboration between carriers, enhance relationships with customers through real time communications.

Transportation applications: With IoT, it is now possible for smart cars to communicate with each other in real time, and exchange data that could improve the safety of drivers, passengers, and pedestrians. For example, a connected vehicle could alert other vehicles of potential road hazards, a reckless driver, or upcoming traffic delays. In addition, drivers of smart vehicles can receive real time information about congested traffic, road closures, and public transportation schedules. This information gives drivers effective route optimization, consequently saving one's time and energy. The Office of the Assistant Secretary for Research and Technology (OST-R), and the United States Department of Transportation (USDOT) is working with local transportation agencies, vehicle and device manufacturers, on mandatory vehicle IoT technology. A study conducted by National Highway Traffic Safety Administration (NHTSA) in 2016 [6] shows that connected vehicles could reduce the occurrence of crashes by 80%.

Healthcare applications: IoT has improved many aspects of the healthcare industry. Bringing the IoT into the medical domain results in patient healthcare that is better, safer, and simpler. IoT has refined the patients' experience with healthcare providers. On one hand, communication through smart devices (e.g., wearable devices) reduces in-person visits and lets patients manage their care from home. IoT increases patients' health awareness as patients become more engaged with health and fitness applications that assist them to analyze their own health data. On the other hand, IoT helps healthcare providers switch from cost-based services to value-based services. The accurate data collection, such as tracking the heart rate, temperature, and blood pressure, allows physicians to spend less time on logistics and more time on treating conditions and consulting with patients. This speeds up the decision-making process, reduces human error, and improves the treatment experience for both the patient and the physician. In addition, IoT devices can help first responders to save lives in emergencies. For example, real time reporting of patient health indicators through mobile applications allows healthcare providers to get information faster and provide higher quality care before the patient arrives at the hospital.

Smart environment applications: The proliferation of sensors and actuators in our surroundings enables a wide range of smart applications that improves several aspects of our daily lives. A recent study conducted by IoT analytics [7] to rank IoT applications based on their popularity shows that, in 2018 the most identified IoT projects were associated with Smart City (367 projects), Industrial Settings (265), and Connected Building IoT projects (193). Smart homes and buildings have recently drawn a great deal of attention although the concept has been around for many decades. Initially the smart home service was stimulated by the automation of domestic systems that aim for comfort, health, safety, reduction of labor, and energy efficiency. Then came the wireless internet and smartphone technologies that extended the concept to services which can be remotely accessed and controlled at any time and from anywhere. However, IoT adds a level of interconnectivity between household appliances such that these appliances can interact to understand the behaviors of the residents and provide intelligent services autonomously [8]. For example, heating can be adapted to personal preferences and to the weather conditions; the lighting can change according to the activity and the time of the day. IoT is changing the focus of building automation from cost reduction to features that enhance the user experience while interacting with the building facilities. For example, modern university buildings now use sensors, actuators and IoT connectivity to enable customisation of spaces in offices, classrooms and conference rooms based on occupancy and occupant preferences.

Despite the aforementioned compelling advantages and benefits that IoT offers, the new technology has not achieved widespread adoption yet. This is primarily due to security and privacy concerns that IoT raises. The omnipresence of IoT devices allows for the collection, processing, and dissemination of sensitive information about individuals, which undermines their security and intrudes on their privacy. The over-provisioning of access permission to the data generated by IoT devices and advancements in data analysis techniques has opened the door for third parties such as Data Brokers [9] to aggregate, infer and release sensitive information about the administrators of these devices. For example, collecting real time energy data at fine granularity allows the electric utility company to study the consumption patterns and improve customer experience. However, off-the-shelf data analytic tools can reveal sensitive inferences about the behavior of the occupants including daily schedules, usage patterns, and anomalies. Therefore, it has become evident that security and privacy concerns will impede the widespread acceptance of IoT technology, and that access to IoT devices must be limited and controlled. Otherwise, the security and privacy risks of the new technology will outweigh any of its benefits.

Our objective in this research aims at developing a dynamic access control framework that prevents unauthorized access to IoT devices with better adaptability to the highly dynamic IoT environments.

The remainder of this chapter presents the motivations behind our research, summarizes the thesis contributions and outlines the thesis document.

1.1 Motivation

Access control schemes have been studied extensively over the past few decades and remain an area of intense research interest. This is due to rapidly changing technology and growing security and privacy concerns. However, controlling who can access what under which conditions is not a trivial task in the IoT context. The extremely large number of heterogeneous and resource-constrained IoT devices communicating over dynamic, distributed, and ad-hoc networks with low bandwidth connections creates a unique set of authorization and access control challenges, rendering standard access control policies, models, and mechanisms unfit to IoT scenarios [10]. In this thesis, we identify three major limitations in traditional access control mechanisms to IoT environments, manual access policy management, discontinuous access decision enforcement, and static access permission assignment. In the following, we summarize these limitations and their respective challenges:

Manual access policy management: Traditional access control systems and authorization techniques are proposed for closed computing environments where all users and resources of the system are known in advance, and rely on preconfigured access control policies to fulfill the system security and privacy requirements that are unlikely to change during run time. However, the unbounded interactions among IoT devices can result in frequent changes in security and privacy requirements of the resource administrators and consequently frequent changes in the underlying access control policies. For example, it is expected that IoT entities will be joining and leaving the system anytime and at their discretion. In such environments, manual policy management such as adding/removing access rules, identifying and resolving conflicts in access control policies, becomes a complex and error-prone task. Indeed, the system administrators have to maintain correct, consistent and up-to-date access control policies in order to keep pace with frequent changes in security and privacy requirements of the highly dynamic environment. Otherwise, incorrectly configured, conflicting or obsolete access control rules can expose system resources to unauthorized access. Therefore, automation of the access policy generation process is required to overcome inflexibility in traditional policy management techniques, eliminate errors and conflicts in access control policies and ensure authorized access in highly dynamic IoT environments.

Discontinuous access decision enforcement: Traditional access control models are proposed to protect data that is permanently stored with static or infrequently changing access control policies. Typically, these models enforce access decisions only when time access is requested and do not consider changes in access conditions while the resource is in use. Advancements in IoT enabling technologies along with ubiquitous connectivity have led to a new generation of smart services based on real-time data access. The popularity of ubiquitous data access and accelerated adoption of these services pose significant challenges on user and data privacy. Thus, controlling access to such services in highly dynamic environments with continuously changing context becomes even more challenging. The wide adoption of IoT in our everyday life in many domains such as healthcare and military operations requires continuous and tight access control to prevent unauthorized and unintended access. A delay in making access decisions when context changes may result in consequences that cause harm and damage. Therefore, continuity in access policy enforcement becomes a necessity in highly dynamic IoT environments not only at the time of request, but rather for the entire access session.

Static access permission assignment: Typically, access control models assign access permissions based on static considerations such as identity or roles. However, these models usually result in defining access policies that assign more access permissions than required by the resource requester. To alleviate the problem, researchers have focused on improving the expressiveness of access control policies such that policy administrators can define customized access control policies that consider, beside the identity and role, the dynamic access condition factors such as time, location, purpose of access, and interrelationships among users. The access control literature refers collectively to these factors as access context. However, the highly dynamic nature of IoT makes it difficult, if not impossible, to expect all access contexts and assign the appropriate access permissions to each context. Such unique characteristics of IoT requires an adaptive permission assignment technique that adjusts access control policies at run time in order to prevent exploitation of out dated access control policies and reduces the chance for users to abuse or misuse excessive access permissions.

1.2 Problem Statement

Traditional access control approaches have failed so far to consider continuity in access decision enforcement and to provide the necessary access control infrastructure that facilitates continuity in access policy enforcement and adaptability to unexpected access contexts. Although many basic principles of standard access control models continue to apply in IoT contexts, still it is not clear how to assign the appropriate level of access permissions to resource requestors while there is a degree of uncertainty surrounding the access contexts under which the protected IoT resource can be accessed. This research provides a dynamic access control framework for IoT environments. The framework provides an adaptive permission assignment technique that adjusts the access control policies based on both proactive and reactive contextual measures. In addition, the framework provides a responsive access control infrastructure that accommodates potential policy adjustments at run time. The infrastructure integrates an automatic policy specification[1] that generates, on the fly, context-aware and conflict free access rules, and a continuous policy enforcement component that keeps pace with changes in access policies as well as access context to maintain authorized access while the resource is in use.

1.3 Objectives

The purpose of this research is to answer the following questions:

1. What are the attributes of an effective access control approach for expressing, applying, and continuously enforcing individual's security and privacy preferences in highly dynamic IoT scenarios?

2. Can we simplify the process of access policy specification and alleviate the administrative efforts associated with manual management of access control policies in IoT?

3. How best to cope with frequent changes in access contexts in IoT environments?

4. Where to implement the access control logic to support continuous access control enforcement and real time interactions in a distributed and resource limited environment such as IoT?

[1]In this thesis, the term "policy specification" refers to the process of translating high level security and privacy requirements written in natural language into a formal format that can be interpreted and executed by computer systems.

5. Can we reason about the user's access patterns and adapt access control policies to unexpected access behaviors?

1.4 Contributions

This thesis answers the aforementioned research questions and contributes to the field of access control as follows:

- We provide an extensive analytical study of proposed access control solutions for the IoT environment. We also provide a formal definition of dynamic access control and built our classification taxonomy based on this definition. We evaluate the access control solutions quantitatively against the IoT access control requirements in general and the IoT dynamic nature in specific. The study highlights the merits and limitations of existing access control approaches in IoT context and drives the rest of thesis contributions.

- We introduce an automatic policy specification method that simplifies the access policy specification process, reduces errors in access policies administration and produces conflict free access rules. To overcome the rigidity in policy specification methods, we breakdown access control rules to the fundamental elements: subject, object, and operation. For each element, we describe the context under which it can associate with other elements. We call this context the *Guard Context*. Finally, we store the GCs as primitive facts in the policy database. Access is granted, if there exists in the policy database a GC that associates the requesting entity, requested resource, and operation, and that the operational context of these elements satisfy this GC. Otherwise, access is denied by default. Our approach allows for automatic generation of access control policies in less time and space complexities. In addition, it obviates the burdens of detection and resolution of access policy conflicts because it does not store the access policies in the final form. A change in a primitive fact will automatically affect all access control policies that can be generated based on this fact.

- We design and implement a continuous access enforcement mechanism that is sensitive to real time changes in access contexts. This component ensures authorized access not only at the time access is requested, but rather for the entire access session. In our design, we continuously monitor two types of changes in the access context: the session operational context as well as the GCs that represent the access policies or rules that grant the access session. If the operational context of the session violates the GC, it results in immediate closure of the access session. However, if the GC changes, this results in re-evaluation of all ongoing sessions. We quantify the time complexity of the re-evaluation process in an IoT like environment. Results show that our approach can efficiently enforce access decisions in highly dynamic and resource constrained IoT environments.

- We design and implement an adaptive permission assignment mechanism that allows for dynamic adjustment of access control policies based on device behavioral patterns and access contexts. The mechanism assesses the compliance of an IoT device to normal access behaviors and uses proactive measures to generate controls that prune access control policies at run time. With this, we narrow down the access permissions of misbehaving IoT devices and reduce the risk for IoT users[2] to exploit obsolete access control policies.

- We propose a generic access control framework that integrates all algorithms, models, and mechanisms developed in this research work. The framework will provide robust access control functionality including automatic access policy specification, continuous access policy enforcement, and adaptive access permission assignment.

1.5 Organization of Thesis

The remainder of this document is organized as follows. Chapter 2 presents the background and literature survey in access control policies, models, and mechanisms. Chapter 3 presents DACIoT, our proposed dynamic access control framework for IoT environments. DACIoT consists of three components: an automatic policy specification schema, continuous access policy enforcement mechanism and adaptive access policy adjustment technique. The automatic policy specification schema of DACIoT focuses on simplifying the access policy management process because IoT paradigm shifts this responsibility from a skilled and well experienced policy administrators to entry-level IoT users. The schema is centralized around the concept of context. An IoT device administrator only needs to define the primitive facts that describe the context under which their devices can be accessed, and have the access engine match these facts automatically into conflict-free access rules at run time. The continuous access policy enforcement mechanism supports context-aware and real-time decision-making. When a policy change is detected while a resource is in use, the

[2]In this thesis, we use the term "IoT user" to refer to users who control IoT devices either physically through direct interaction or remotely via IoT application.

mechanism re-enforces the updated policy to ensure continuity of authorized access to the resource for the entire lifetime of an access session. The adaptive access policy adjustment technique reduces the uncertainty of access contexts in IoT environments; it handles access scenarios where authorized IoT users, accidentally or intentionally, misuse their access permissions. Our approach to policy adjustment adopts machine-learning techniques to detect such access scenarios and adjusts the underlying access policies accordingly. DACIoT lays the foundation for effective and efficient dynamic access control solutions that take into consideration automation of access policy, continuity of access policy enforcement and adaptability to changing access contexts.

Chapter 4 presents the experimental validation and performance evaluation of DACIoT. The chapter starts with a use-case access scenarios to university resources. The use-case proposes different access scenarios that capture the most important access control requirements in highly dynamic IoT environments. Three metric are used to validate DACIoT framework including the correctness, consistency and completeness of DACIoT access control policies. Many experiments are conducted to evaluate the performance of DAIoT using an IoT prototype and based on real-life dataset. Results show that DACIoT provides improved security, dynamic adaptability and sufficient scalability to IoT environments. Chapter 5 highlights the main issues addressed in this thesis and outlines future research directions.

Chapter 2

Background and Literature Survey

2.1 Introduction

In the era of IoT, it has become possible for a set of smart devices to collaborate autonomously and communicate seamlessly to achieve complex tasks that require a high degree of artificial intelligence. Using computers, sensors and networks in controlling remote objects and appliances is not a new concept and has been around for decades. For example, in 1999 an Internet conference featured the first remotely controlled toaster over the internet [11]. Other examples of early Internet-connected objects included the soda machine at Carnegie Mellon University [12] and the coffee pot in the Trojan Room at the University of Cambridge [13].

What makes IoT a new concept is the advancements and convergence of a variety of IoT enabling technologies that make it possible to interconnect a larger number of smaller devices at a low cost [14, 15, 16, 17]. While ubiquitous connectivity via global adoption of IP-based networking enables everything to be cheaply connected and globally addressable, miniaturization allows for advanced computing and communication technologies to be embedded into very small objects. Furthermore, advances in data analytics and cloud computing services allow distributed and resource-constrained IoT devices to harness knowledge and information from powerful back-end computing and analytic capabilities.

Various forecasts have projected the potential number of IoT-connected objects in the near future; all indicate a significant and increasing growth of IoT devices. According to Cisco, there will be more than 24 billion Internet-connected objects by 2019 [18]. Morgan Stanley [19] forecasts a higher number: 75 billion network-connected objects by 2020. While Huawei estimates an even higher number of 100 billion IoT connections by 2025 [20].

The increasing number and density of IoT objects in our surroundings, along with the fast developments in the integrated technologies are widening the range of potential IoT applications in several domains. Many research organizations and companies have defined their own taxonomies and classification schemes to categorize the IoT applications. For example, McKinsey Global Institute [21] classified IoT applications into ”settings” which represent the expected value that IoT can create for both users and industry (e.g., human: wearable devices, home: energy management, office: security systems). Other applications [22] focus on IoT device types (e.g., wearable and appliance). While [23] categorizes IoT applications from location-based implementation perspective (e.g., smart home, smart city). Regardless of its application domain, categorization and purposes, IoT is expected to greatly impact the way we live and the way we carry out our everyday tasks.

With all the convenience that IoT brings to us, security and privacy remain major concerns to stakeholders. In their 2015 report on privacy and security in IoT [24], the Federal Trade Commission (FTC) stated that privacy and security are core requirements in IoT and that concerns among IoT users could undermine the user's confidence of the new technology and impede its widespread adoption. Considering the vast amount of personal data that IoT collects and uses, a robust access control system is imperative to ensure that only authorized parties can read sensor data or command an actuator.

In this chapter we explore the unique security, privacy and trust challenges of IoT. Then we provide a background on the basic access control concepts followed by a comprehensive review of standard access control models pointing out their strengths and weakness. Next, we identify the IoT-specific access control requirements, provide a formal definition of dynamic access control and introduce our classification taxonomy. After, we survey the access control solutions proposed for IoT, presenting in depth analysis to the major access control challenges each solution addresses, and evaluating the latest research efforts from dynamic authorization perspective. Finally, we identify three main attributes that should be intrinsically supported by IoT oriented access control systems.

2.2 Security, Privacy and Trust Challenges in IoT

Security, privacy, and trust are crucial and primary requirements for widespread adoption of any new technology. However, IoT features a unique set of attributes that make it challenging to fulfill these requirements. Traditional protection mechanisms such as standard access control approaches, secure protocols and privacy assurance are not enough in the IoT context, and need to be carefully analyzed to decide whether to integrate them into IoT as is, adapt them, or entirely redesign new ones that are a better fit. This section highlights unique security, privacy and trust challenges that present in IoT environments.

2.2.1 IoT Unique Security Challenges

The highly dynamic nature of IoT environments poses major security challenges in different ways compared to traditional Internet.

- Unlike the traditional Internet, computing in IoT environments is ubiquitous and can take place at anytime and in any location using any form of IoT devices. The rapid mobility of IoT devices increases the uncertainty of the contexts (e.g., location) under which the data generated by a device can be accessed. For example, wearable devices such as fitness trackers usually connect to the Internet through gateway devices (e.g., smartphones or wi-fi access points). These gateways continuously send the health data reported by wearable devices to the cloud for further analysis. The interactions among these devices typically take place over short-range wireless technologies such as Bluetooth and Wi-Fi. While not usually enabled by default, most wearable devices have settings that allow connections to be initiated automatically to other devices in their proximity without notifying users. For instance, connecting the fitness tracker to a free Wi-Fi network exposes user's information to cyber-attacks including access to their online accounts and credentials.

- Many IoT solutions and applications are designed to be deployed on a large scale. The sheer number of IoT devices and users will result in an unpredictable number of dynamic and autonomous interactions, rendering current Internet security tools, methods and strategies inadequate for such deployments. For example, security mechanisms that are considered adequate at the IoT deployment time might become obsolete or insufficient for evolving security threats in later stages of the lifecycle of an IoT device. This vulnerability may happen in IoT scenarios when upgrading and reconfiguring IoT devices is inapplicable due to lack of support or discontinuity.

- Some IoT deployments may employ devices that have identical or common characteristics; a single security vulnerability in one device can easily propagate to an extremely large number of IoT devices that share the same characteristics. Unlike the traditional Internet, large-scale attacks on IoT devices can extend to the physical world and cause a mass chaos. For example, the December 2016 cyberattack on the Ukrainian power grid. The hackers planted a malware in the network of Ukraine's national grid operator. The malware was designed to open every circuit breaker in a transmission station north of the city of Kiev, blacking out a portion of the Ukrainian capital equivalent to a fifth of its total power capacity. Later analysis of the malware code revealed that the malware can be adapted to different Internet connected IoT devices and can be easily reused or launched simultaneously to attack multiple targets.
- Many IoT devices have no way to alert users when security breaches arise. Therefore, it is expected in IoT environments that some security breaches exist and continue for a long time before being observed or resolved.

2.2.2 IoT Unique Privacy Challenges

The privacy concept defines the rules under which data referring to individuals may be accessed [25]. In their June 2015 report on consumer perceptions of privacy in IoT [26], Groopman et al. state that *" Consumers are highly anxious about companies sharing their data: 78% of consumers are highly concerned about companies selling their data to third parties."* They also state *"While older generations show higher concern, strong discomfort with the use and sale of connected device data is pervasive across all age groups, including millennials."* In fact, the data collected, transmitted, stored, and shared by IoT devices is highly personal and may reveal sensitive information such as financial, religious, health, shopping preferences, thereby raising a number of serious concerns and risks around privacy and data access privileges. In the following, we provide a non-exclusive list of IoT specific privacy challenges:

- Discoverability: is a key component to achieve scalability in IoT settings, however, many private IoT things may prefer to remain undiscoverable. For example, having a connected medical device (e.g., heart monitor) in your home can be useful to your primary care, but you may not want anyone else to know.
- Data collection: in many IoT scenarios, users have no knowledge or control over the way their personal data is being collected and used.
- Heavy instrumentation: the proliferation of sensors and smart devices in our surroundings (e.g., homes, cars and wearable devices) increases the sensitivity of the data that is being generated, shared and collected. Therefore, when these devices are connected they effectively allow third parties to digitally monitor our daily activities. For example, sharing the real time energy usage data can provide sensitive information about our presence and activities in our homes.
- Context changes: IoT creates privacy-threatening scenarios where one thing might unexpectedly or asynchronously use data generated by another thing.

For example, consider an employee who has particular access privileges to certain resources within his organization based on his current role. When his role changes (e.g., demoted or retired or some of his responsibilities have been re-assigned to another co-worker), a data privacy violation may occur due to a delay in capturing the privileges of the new role, which might provide the employee unauthorized access to sensitive data.

2.2.3 IoT Unique Trust Challenges

Trust represents the level of confidence and the assurance that one entity (i.e. Trustor) has on another entity (i.e., Trustee) to behave as expected in performing an action within a certain context. In this regard, IoT entities negotiate and exchange some form of credentials before they can trust each other and share knowledge among themselves. Two approaches have been investigated to form trust relationships between IoT entities [27]:

- Direct trust approach: in which trust relationship is formed and augmented by the two interacting entities after they have performed transactions with each other.
- Indirect or third party trust approach: in which a trust relationship is formed based on third party recommendations where interacting entities might not have had any previous transaction before.

The trust concept is characterized as subjective, dynamic, asymmetric, intransitive and context-dependent [28]. Trust covers other important concepts such as security and privacy. The three concepts are strongly related and significantly influence the acceptance and adoption of the emerging technologies such as IoT. However, IoT poses new issues with regard to trust:

- IoT changes the quantity, quality and granularity of data it collects. Therefore, if the large volume of collected data are not trustworthy (e.g., malicious sensor input or imperfect sensor readings), the quality of IoT service may be greatly impacted and not accepted by users.
- IoT services are based on data processing, analysis, and mining. Therefore, trustworthiness must be ensured at this level as well, which requires an efficient, accurate, secure, privacy-preserved and reliable data processing and analysis. For example, the privacy breaching implied in context-aware and personalized services is one challenge to achieving trust in data processing, as users need to disclose and share personal information (e.g., identity, location) in order to enjoy smart services.
- The dynamic and fully distributed nature of IoT environments requires a shift in trust management from the traditional, centralized, and static approaches such as KeyNote [29] and TrustBuilder [30], to fully distributed and dynamic solutions where system entities such as IoT users and devices have no predefined trust relationships.
- Successful trust management has to meet the scalability requirements that IoT requires at different levels, including identity management and service provisioning.

2.3 Access Control

A vital component of any system or environment is security and information access control, and IoT is no exception. IoT needs to regulate access to protected resources (data and devices) from unauthorized access and privacy leakage. Therefore, Access Control (AC) is becoming an increasingly important security measure in IoT context. This section explores the basic concepts, phases, and elements of the AC systems.

2.3.1 Access Control Concepts

AC is a combination of four security concepts, which are the identification, authentication, authorization and accountability.

- **Identification**: is the process where a user gives a valid and recognized credentials to the system (e.g., username and password, smart card, retina scan and so on).
- **Authentication**: is the process by which a system verifies the identity of an accessing entity. Typically, the process is performed by asking accessing entities to provide their credentials. The accessing entity is authenticated when the provided credentials are valid and accepted, otherwise, authentication fails [31]. Nevertheless, the authentication process does not determine what actions an accessing entity can perform or what part of system resources they can access.
- **Authorization**: is the process that determines whether an authenticated entity has sufficient privileges to access system resources and what operations are allowed or prohibited for this specific entity on the resources of interest. The

process enforces the system security and privacy requirements on protected services and resources [32].The level of authorization that an entity can be assigned is determined by evaluating its associated properties, such as identity, roles, proximity, access history, privacy preferences, and group membership (e.g., administrators or customers), against a set of predefined access rules.

- **Accountability**: is the process that guarantees that all operations carried out by individuals, systems or processes can be identified and that the trace to the subject and the operation is maintained [33].

2.3.2 Access Control Phases

There are three primary phases through which access control systems are developed:

- Phase I: determines the high-level security and privacy rules according to which access control must be regulated. The collection of these rules referred to as access policy (e.g., HIPAA) [34].
- Phase II: is the formal representation of the access policy and procedures, referred to as access control model (e.g., Role Based Access Control Policy (RBAC)) [35].
- Phase III: is the development of the low-level software and hardware functionalities that implement and enforce the security rules defined in the access policies and formalized by the access control model. Typically, these functionalities are referred to as policy enforcement mechanisms (e.g., antivirus software and firewalls).

2.3.3 Access Control Elements

These are the core elements of the access control upon which functions of the access control mechanism enforce the access rules defined in the access control policy. Access control elements include:

- Subject: is the entity that actively causes information to flow between system components or that changes the system state (e.g., person, device, application and process).
- Object: is the passive entity that receives data (e.g., light bulb and door lock).
- Operation: is the action invoked by a subject and applied to an object (e.g., set and get).

Access *privilege* or *permission* are two access control terms that are used interchangeably in the literature and refer to the set of operations a subject can perform on a certain object. However, permission is a property of an object that implies consent has been given to a subject to perform an operation on that object, while privilege is a subject property that lets the subject perform the operations that are not ordinarily allowed on a certain object.

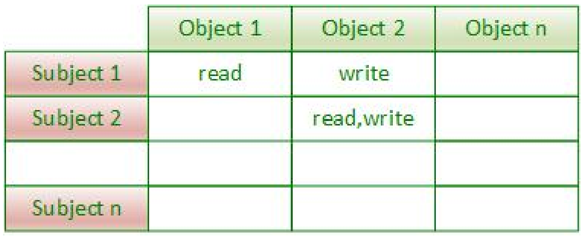
2.4 Basic Access Control Policies and Models

In current literature access control policies are divided into two categories the Discretionary and Mandatory access control policies. In this section we discuss the access control models that fall under each category and summarize their strengths and weaknesses.

2.4.1 Discretionary Access Control Policies

Discretionary Access Control (DAC) [36] controls access to an object based on access control policies that are determined by an object's owner, group and/or subjects. DAC uses the user identification supplied during authentication to define the controls that grant and restrict access to the object. DAC is discretionary because the object's owner determines the object access permissions. Therefore, DAC allows an owner of a certain object to delegate their access rights over that object to other subject(s). For example, a subject (e.g., Alice) who has access rights (e.g., read) to a shared folder on a server can leverage the flexibility of DAC and easily delegate this access right to another subject (e.g., Bob) by creating a username and password for Bob. In addition, DAC provides fine-grained and coarse-grained access control decisions, both per-user and per-group respectively. However, the main issue in DAC is the lack of control over information flow. For instance, once Alice passes her access rights on some information to Bob, Alice no longer has control over the information since there is no guarantee that Bob will not pass this information to anyone else. Therefore, DAC is a suitable access control policy for applications whose main concern is information sharing, not strict access. Another important issue associated with DAC is that it does not distinguish between users and subjects (e.g., a process). Therefore, DAC policies are vulnerable to a Trojan horse attack. For example, Bob can send a seemingly useful program to Alice, which hides an illegal function that can delete Alice's files; the hidden function will inherit Alice's identity and bypass the access control system.

Figure 2.1 shows the Access Control Matrix (ACM); the earliest DAC model used for computer systems. ACM defines access rights in a form of matrix where each row represents a subject and each column represents an object. Each cell in ACM



	Object 1	Object 2	Object n
Subject 1	read	write	
Subject 2		read,write	
Subject n			

Figure 2.1: Access Control Matrix.

specifies the access rights a certain subject has over a particular object.

Access list and access capability are the two well-known implementations of ACM that lay the foundation of the two discretionary access control models: the Access Control List Model (ACL) and Capabilities Based Access Control Model (CapBAC). In the following we discuss these models and highlight their strengths and weaknesses

1. **Access Control List Model**

   In the implementation of ACL model, the ACM is stored by columns (i.e., objects), where each object is associated with a list that specifies, for each subject, the set of access permissions on this object. The advantage of this object-centric implementation is that it is easy to grant and revoke access permissions per object. However, determining the access permissions for a particular subject requires every ACL to be checked against this subject. This makes it difficult to figure out which objects a user has access to, hence access revocation is usually done by deleting a user rather than deleting ACL entries. Therefore, ACL is not suitable for large scale and open environments. Figure 2.2 depicts the structure of ACL model.

   The traditional UNIX system of users, groups, and read-write-execute permissions is a typical example of ACL implementation of DAC. The UNIX file mode

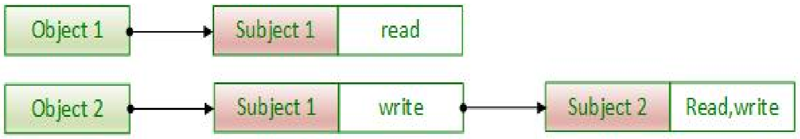


Figure 2.2: Access Control Lists.

Table 2.1: Strengths and Weaknesses of ACL Models

Strengths	Weaknesses
- Simple implementation. - Supports least privilege principle. - Fine-grained access control.	- Does not control information flow. - Vulnerable to Trojan horse attack. - Complex policy management. - Does not scale. - Context insensitive. - Closed world environments.

defines the read, write, and execute permissions for each user, group and others. File ownership is an important component of UNIX that provides a secure method for storing files. Every file in UNIX has the following access levels: owner permissions that determine what operations the owner of the file can perform on the file; group permissions that determine what operations a user, who belongs to the group that the file belongs to, can perform on the file and world permissions that determine operations all other users can perform on the file. Table 4.1 summarizes the strengths and weaknesses of the ACL model.

2. **Capability Based Access Control Model**

In the implementation of CapBAC model, the ACM is stored by rows (i.e., subjects), where each subject is associated with a capability that specifies the access rights of a subject over a set of objects in the system. Figure 2.3 shows the structure of CapBAC model. The key concept of CapBAC is the access token. Tokens can be defined as communicable unforgeable self-contained ticket

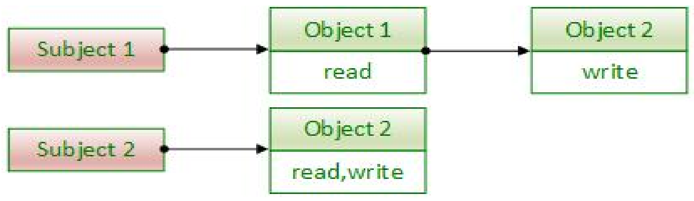


Figure 2.3: Capability Based Access Control Model.

of authority [37]. Tokens contain the information required for their holder to access a resource. The token contains information such as the reference of the resource, and the set of access permissions that the holder possesses to interact with the resource. CapBAC has the following advantages over ACL: it supports delegation of access permissions, where a subject can use the access token to pass their access rights, including the right of delegation, on certain object to other subject(s); it is secure to Trojan horse attack because permissions are assigned to subjects rather than objects; it supports interoperability as it uses standard extendable capability representation using XML-based languages. Despite the promising features in capability-based approach, the complexity of the access token assignment and revocation on a specific resource presents a major drawback that needs to be carefully considered in large-scale computing environments [38].

Microsoft Windows operating system is an example of systems that use access capabilities or tokens to control access to system hardware and software resources. For instance, when a user logs on and is successfully authenticated, the system creates an access token that contains security identifiers that identify the user's account and any group accounts to which the user belongs and a list of access privileges assigned to the user or the user's group. The system uses

Table 2.2: Strengths and Weaknesses of CapBAC Model

Strengths	Weaknesses
- Simple implementation. - Supports least privilege principle. - Supports flexible delegation (can scale) - Vulnerable to Trojan horse attack. - Fine-grained access control. - Interoperable	- Does not control information flow. - Context insensitive. - Complexity in token distribution and revocation.

this token to identify the associated user when a process tries, on behalf of the user, to access a protected object or perform a system administration task that requires access permissions. However, Windows operating system uses what is called *"security descriptors"* to maintain the security information about system's objects. The descriptor contains two types of ACLs: A discretionary access control list that identifies the users and groups allowed or denied access to the object and a system access control list that controls how the system audits attempts to access the object. Table 4.2 summarizes the strengths and weaknesses of CapBAC model.

2.4.2 Mandatory Access Control Policies

Unlike DAC, Mandatory Access Control (MAC) [39] policies are not specified at the discretion of the user, MAC controls access based on access rules defined by a central authority that is aware of the whole system security requirements, therefore, data users neither can delegate access privileges nor they can change access rights. MAC policies include:

1. **Multilevel Security Model (MLS)**

   MLS control access by classifying the system's subjects and objects according to varying security levels. Each object is assigned a sensitivity label while each subject is assigned a clearance label. Access to a particular object is only granted to subjects with a clearance level that satisfies the object's sensitivity level. For example, consider a system which maintains three security levels (Confidential (C), Secret (S) and Top secret (TS)). According to MAC, two principles should hold:

- No-Read-Up: a subject who is cleared to security level C should not be able to read a file with sensitivity level S or TS.
- No-Write-Down: a subject who is cleared to security level TS should not be able to write to a file with sensitivity level S or C.

Figure 2.4 shows the access privileges and permissions for subjects and objects at the TS security level only.

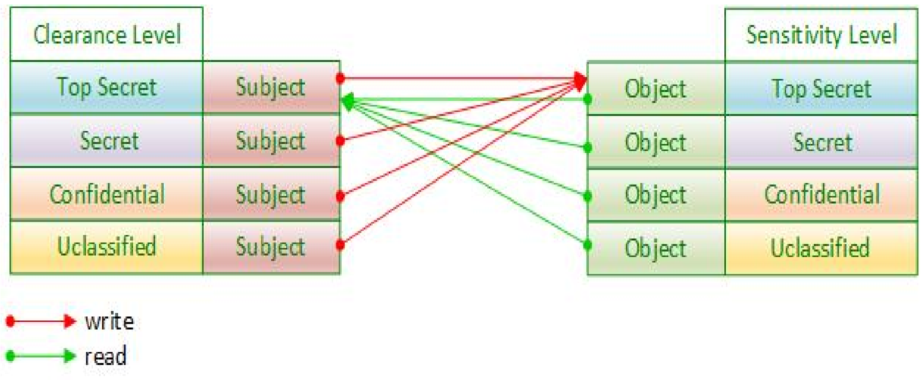


Figure 2.4: Mandatory Access Control Model.

One advantage of MLS is that it controls the flow of information. MLS prevents information flow from higher security levels to lower security levels through the satisfaction of the No-Read-Up and No-Write-Down principles. Another advantage of MLS is that it distinguishes between users and subjects. Therefore, MLS systems are secure to Trojan horse attacks. Although, MLS is designed for applications with tight and controlled information flow, it only controls information flows in legitimate channels. Therefore, MLS has no control over information flows that may happen in covert channels (i.e., illegitimate channels); those channels through which lower security levels can deduce, by inference, higher security level information. MAC is a rigid access control system this is of concern for two reasons: (1) adding new subjects and/or objects requires careful assignment of security classifications, which may not be feasible in practice (e.g., large-scale organization); (2) access is only evaluated based on static security classifications where other runtime factors (i.e., Context) are not considered.

Bell-laPadula confidential model [40] and Biba integrity model [41] are examples of MLS model . Bell-laPadula maintains the two security rules of MAC (i.e., No-Read-Up and No-Write-Down) to preserve confidentiality by preventing data to move down to lower confidential security level. In contrast, Biba maintains data integrity by reversing Bell-laPadula rules: no read down; no write up to prevent moving data from lower integrity levels to higher ones. As MLS is mainly developed to be used in applications where confidentiality is more important than integrity, Biba integrity model has little influence on the development of MAC model. Tabel 4.3 summarizes the strengths and weaknesses of MAC models.

2. **Role Based Access Control Model**

Diversity of practical requirements in different application domains has led to the need for an access control policy that takes advantage of the traditional

Table 2.3: Strengths and Weaknesses of MAC Models

Strengths	Weaknesses
- Controls information flow. - Supports least privilege principle. - Secure to Trojan horse attack.	- Coarse-grained access permissions. - Complex implementation. - Complex policy management. - Context insensitive. - Assumes closed environments.

access models and overcomes their weaknesses. For example, commercial enterprises require an access policy that assigns access permissions to user or group as in DAC policies, and still can define restriction on these assignments as in MAC policies. The Role Based Access Control model (RBAC) [35] came to satisfy these requirements. RBAC is built around the concept of role(s), which reflects the natural structure of an organization and defines the set of duties and responsibilities of the users to whom these roles are assigned. RBAC has the following advantages:

- RBAC simplifies policy management by breaking down the user-permission assignment into two parts, user-role, and role permissions assignment. This allows for grouping users with similar access needs by role (a many-to-one relationship) such that access permissions on a certain resource can be set or revoked once for a single role instead of individual users. (see Figure 2.5)
- Hierarchical authorization : RBAC allows for hierarchies and inheritance of permissions in which restrictive permissions override general ones.
- Least privilege: based on their roles, users always have the minimum access privileges required to complete a task.

- Separation of duties: according to this principle, a user should not have enough privileges that they can exercise in a manner that negatively affect objectives of the resource's administrator.
- Access views: RBAC allows one user to have multiple roles (a one-to-many relationship) according to their duties within the organization such that the user can have multiple levels of access or views on the same resource.

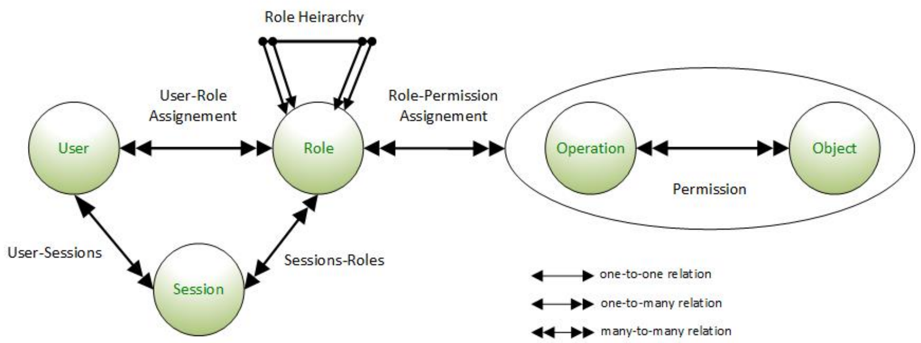


Figure 2.5: Role Based Access Control Model.

However, modern organizations have different types of users with different access needs, this introduces many administrative and policy enforcement challenges in RBAC. For example, grouping users into roles makes it difficult to define granular access control for individual users. Many users have one-of-a-kind set of access permissions, which usually results in creating additional specific roles to exclude specific users who fall into a particular role, but do not necessarily need to have the same permissions granted to other members of the role. This solution usually results in a number of roles that is equal or greater than the number of users; a RBAC dilemma known as role explosion. In addition, abstracting access control policy into roles does not always express the real world access requirements and cannot handle contextual factors that are relevant to access control decisions (e.g., time and location). Also, these roles should be known in advance and that both user-role and role-permissions should be statically assigned, which is not always possible in real world access scenarios, especially in highly dynamic and open environments. This adds difficulties to the system setup and access policy management, therefore, RBAC does not scale. Furthermore, managing access policies for multiple application domains, each of which requires different access control policies, is challenging in that the same role may require to be assigned different access permissions in different domains.

Azure [42] is Microsoft's cloud computing platform and example of services that uses RBAC to control access to its resources. The RBAC model is built on the Azure resource manager and enables subscribers (i.e., users and applications) to manage resource (e.g., virtual machines and networks) in a subscription. Based on supported access level, RBAC allows users or a group of users to manage one resource or group of resources in a subscription. Table 4.4 summarizes the strengths and weaknesses of RBAC model.

Table 2.4: Strengths and Weaknesses of RBAC Model.

Strengths	Weaknesses
- Simple policy management. - Supports separation of duties. - Support DAC and MAC.	- Complex system setup. (role design and engineering) - Critical Scalability in open world environments. - Critical interoperability in collaborative environments

3. **Attribute Based Access Control Model**

Attribute Based Access Control (ABAC) [43] is a promising alternative to all aforementioned access control models. It encompasses the benefits of prior models, as well as it brings new features suitable for dynamic and open environments. ABAC controls access to protected resources based on:

- A set of arbitrary attributes associated with the user. User attributes can be static (e.g., identity, role, etc.) or dynamic (e.g., age, location, trust level).
- A set of arbitrary attributes associated with the object, which include attributes about the metadata related to the object (e.g., manufacturer, default settings, CPU capabilities, memory size), or the content of the object based on application (e.g., temperature readings of a thermometer, location coordinates from a GPS, status of a smart switch).
- The environmental conditions that are relevant to the current state of the system (e.g., current time of day, date, number of users logged in).
- Administrative attributes, these are configuration attributes that apply to the whole system and are either manually set the by administrator or by some automated process (e.g., threat level, minimum trust level, maximum access frequency, maximum session length).
- A set of policies that are specified in terms of those attributes and conditions. (see Figure 2.6).

ABAC has many advantages over previous access control models. It focuses on attributes instead of identity or roles, and controls access by evaluating rules against a wide range of attributes. This makes ABAC a preferred model for

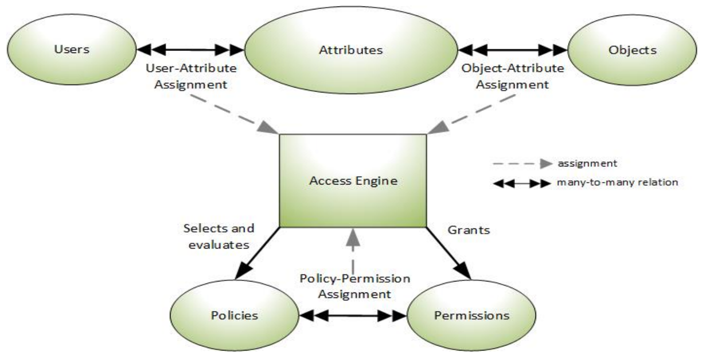


Figure 2.6: Attribute Based Access Control Model.

highly dynamic and open environments. First, ABAC simplifies policy management since there is no need to associate access permissions directly to users, roles or groups. ABAC policies are flexible in the sense that it creates access rules without specifying individual relationships between subjects and objects. Second, ABAC allows unknown subjects to request access to multiple objects with no upfront knowledge. Therefore, ABAC policy can scale easily and subjects can join and leave the system without the need to modify the underlying access control rules. Third, ABAC supports dynamic and real time decision-making, where changes in attributes immediately affect access decisions between access requests and at runtime. Lastly, ABAC supports fine-grained access controls since access permissions are tailored to the user access needs based on their attributes.

One disadvantage of ABAC is the complexity resulting from the heterogeneity of user and object attributes, which incurs added cost to maintain all attributes in a standard format. Attribute standardization, however, could result in low expressiveness and less flexibility in access policy specification. Therefore, ABAC policies are limited by the richness and availability of attributes [44].

ABAC policies have proven applicability to various levels of technology stack and an organization infrastructure. For example, firewalls can use the attributes of a packet's source and destination (e.g., IP address) to control the traffic in and out an organization web server. For the database security, ABAC can define an access policy, for instance, that allows managers to see database transactions only in their region. The same concepts can be extended and at the data layer level ABAC policies can control access to data at the table, column, field, cell by defining filtering conditions and masking based on attributes. For example, HeABAC [45] is an extension to the authorization capabilities offered by Hadoop [46] core and other ecosystem projects, specifically Apache Ranger [47] and Apache Sentry [48]. HeABAC uses ABAC policies to control access to Hadoop's big-data resources at fine granularity. (i.e., folder, sub-folder, file, sub-file, and other granular). Tabel 2.5 summarizes the strengths and weaknesses of the ABAC model.

Table 2.5: Strengths and Weaknesses of ABAC Model.

Strengths	Weaknesses
- Fine-grained access control. - Support multiple policies DAC, MLS and RBAC. - Scalable. - Context sensitive.	- Complexity in attribute authentication and standardization.

4. **Usage Control Model.** Traditional access control models focus protecting data objects only at the server side and have no control on the usage of these objects at the client side. Usage Control (UCON) [49] unifies the objectives of traditional access control approaches and goes beyond them covering other aspects such as privacy and trust. Similar to ABAC, UCON controls access

based on attributes of the subject and object as well as contextual attributes. Therefore, it UCON has the inherent advantages of ABAC model in terms of flexibility, scalability, fine-grain access control, and adaptability to open and dynamic environments. UCON is built on the following components: Subject, Object, Rights, Authorization Rules, Obligations, and Conditions. The main advantage of UCON over traditional access control policies is that it controls access with and without the existence of a central authority (i.e., server side access control, client-side access control or both).

In order to meet the requirements of open network environments, UCON introduces two new features to the access control, one the mutability of attributes and two, continuity in access decisions. These features are at the core of UCON models. For instance, attributes in UCON are updated as a side effect of the usage process rather than administrative ramifications. Therefore, if access attributes change while access is in progress, UCON will revoke access permissions and terminate the access session. Thus, access enforcement continues after the execution of access decisions. The high mutability in attributes and continuity in access decision-making make UCON a recommended choice for highly dynamic and open environments. However, UCON is only a conceptual model, and still lacks precise definitions of access control process [50]. In fact, it assumes the availability of runtime monitors observe the execution of accessing subjects and ensure that their behavior adheres to an access policy. More research is needed to prove the feasibility of UCON, particularly for resource-limited and time sensitive environments. Tabel 2.6 summarizes the strengths and weaknesses of UCON model.

Table 2.6: Strengths and Weaknesses of UCON Model

Strengths	Weaknesses
- Support DAC, MAC and RBAC policies - Support continuous enforcement using obligation and recommendation. - Privacy aware. - Supports dynamic authorization. - Context sensitive.	- Complex implementation. - Complexity in attribute authentication and standardization

2.5 Access Control Requirements in IoT Scenarios

Confidentiality and integrity are two fundamental objectives of information security. The principle of information confidentiality is to protect the information from disclosure to unauthorized users while making sure that the authorized users can access the information. The principle of information integrity refers to protecting the information from being modified by unauthorized users.

Cryptography plays a key role in the protection of confidentiality and integrity of information. Information encryption ensures that only users who have the decryption key can access the information and potentially modify it. However, common encryption methods typically demand more resources than that can be supported by IoT devices.

Other approaches to ensure information confidentiality and integrity in IoT include using access control techniques that assign access permissions and restrict access to sensitive information based on user entitlement and access contexts.

While there are many platforms and technologies available for IoT, access control issues are often overlooked. In IoT context, controlling who can access which resources and for what reason is rather complex. The extremely large number of IoT devices with low power requirements communicating over dynamic, distributed and ad-hoc networks, create a unique set of authorization and access control challenges, rendering standard access control policies, models and mechanisms unfit for IoT scenarios. Therefore, a robust access control system is needed to ensure that only authorized parties can access sensor data or command an actuator to perform an operation. In the following, we briefly list a non-exclusive set of access control requirements in IoT environments:

1. Scalability: The extremely large volume of users, applications and devices that will be connected to IoT requires an access control model to be scalable in terms of the number of users, devices, and policy decision and enforcement points. However, scalability should not result into complex policy specification and management.

2. Real time authorization: IoT has enabled a wide range of smart services and applications based on real time data access. Unlike the traditional Internet, a delay in making access decisions in an IoT context can result in unauthorized access to sensitive data and consequently cause damage. Access decision-making must be made as close as possible to where access requests originate, this enables the access control system to take advantage of real time access to context information in making access decisions that meet the minimum end-to-end delay requirements of real time IoT applications. In addition, authorized access should be continuously ensured not only at the time access is requested but for the entire access session.

3. Context awareness: In highly dynamic computing environments such as IoT, access decisions should not only depend on who the user is, but also on dynamic

attributes that describe the status of the user, resource and the surrounding environments in which the request takes place. Operational factors (e.g., processors, memories and operating systems), situational factors (e.g., location, time, and network security configuration) and environmental factors (e.g., temperature, humidity) can greatly affect access decisions and consequently the performance of the access control system. In the literature, these dynamic factors typically referred to as context information. Context-awareness is key enabler of dynamic access control and authorization. For example, consider a business application, where users (employees) have different access privileges according to their role. Such privileges are valid only within the enterprise premises, while they may have reduced access privileges outside the buildings. This allows for access decisions to be made dynamically based on the users' location. In addition, as the context may change frequently, access permissions should change accordingly. For example, consider a former student is visiting his supervisor. When the student nears the university campus, based on his current context (i.e., student's profile and location) they will be able to access the Wi-Fi network. As the student's context changes (e.g., the student enters a school building), they will be dynamically assigned more access privileges such as opening the lab door. These simple scenarios demonstrate how the dynamic changes in access context affect user privileges in an IoT like environments. Therefore, dynamic access control mechanisms that handle such situations are inevitably required to unleash the full potential of the IoT technology.

4. Lightweight authorization: IoT devices are often resource-constrained and do

not support complex computations. Energy saving is crucial in IoT environments, especially when access control mechanisms are implemented at the device level. Designing an access control model for IoT should take into consideration limitations in IoT resources and adopt security technologies that do not incur considerable computation or communication overhead.

5. Distributed access control architecture: In the literature, access control solutions are divided into centralized, distributed, and hybrid architectures. Centralized access control approaches allow standard access control technologies such as eXtensible Access Control Markup Language (XACML) [51], Security Assertion Markup Language (SAML) [52], and Hypertext Transfer Protocol Secure (HTTPS) [53] to be used on non-constrained hardware, hence reducing computation complexity at the end-devices. However, over-centralization in access decision-making results in access decisions that do not match the real time context of the requestor, requested resource as well as the environment in which access requests take place. Distributed access control approaches use lightweight technologies such as JavaScript Object Notation (JSON) [54], Constrained Application Protocol (CoAP) [55], IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) [56] and Elliptic Curve Cryptography (ECC) [57] to bring the access logic to the end devices. As such, end-devices are no longer passive entities and can autonomously control access to their resources and realize end-to-end security. However, it is not feasible to implement standard access control models on resource-limited devices as these models require intensive computation capabilities that are, in most cases, not available at these devices. In hybrid access control approaches, the processing is shared across

multiple authorization nodes (i.e., network edge devices or gateways), but the decisions may still be centralized and use complete system knowledge. This allows the access control systems to use standard security technologies for access decisions making while still considering context information in access decision enforcement.

6. Interoperability: In collaborative IoT scenarios, different IoT application domains incorporate and integrate their resources, services and users in order to achieve common goals. Each application domain may implement different access control policies, raising the need for an access control mechanism that hides the diversity of access control policies and supports seamless decision-making for access requests across collaborating domains.

An IoT oriented access control system should intrinsically support the security mechanisms that fulfill the aforementioned technical access control requirements.

2.6 Dynamic Access Control

From the literature on access control, we noticed that most of existing access control solutions and approaches have considered context awareness as a key enabler for dynamic access control. In this respect, researchers have focused on improving the flexibility of access control policies by incorporating context information in access decision-making process; the more contextual factors incorporated in access policy, the more flexible and fine-grained the access control becomes. Typically, researchers in access control refer to these approaches as dynamic and/or context-aware access control models.

2.6.1 Definition of Dynamic Access Control

We argue that context awareness is necessary but not sufficient for dynamic access control; that is all dynamic access control policies are context-aware but not all context-aware access control policies are dynamic. From the system engineering perspective, a system is dynamic if it has the following properties:

- A phase space where elements of the phase space represent all possible states of the system.
- Time, which may be discrete or continuous and may extend into the future or the past.
- An evolution law or rule, which enables us to determine the state of the system at each moment of time from its states at all previous times. The most general evolution law is time dependent and has infinite memory.

A formal definition of a dynamic system can be found in [58]. From the control theory perspective [59], a control system is static if its output depends only on its current inputs; it is also referred to as memoryless control system, while a control system is dynamic if its output depends on the past or future (i.e., predicted) inputs stored in its memory. To the best of our knowledge, none of the proposed access control models and approaches have explicitly provided a formal definition of dynamic access control. Borrowing from system engineering and control theory, we define dynamic access control as follows:

"An access control system is dynamic if its access decision, at any point of time, is dependent on past and/or future context information that can characterize the situation of the core access control elements and influence the future access control

decisions."

Proceeding from this definition, it follows that the evolution rule of a dynamic access control system is a function that describes what future authorizations follow from the current access context. The function can be deterministic, that is, for a given time interval only one future authorization state (i.e., permission assignment) follows from the current access context. However, some access control systems are stochastic or adaptive, in that random access contexts also affect the evolution of the access decision-making and future authorization state.

2.6.2 Classification Taxonomy

In this section we define a classification taxonomy for access control approaches proposed for IoT. The taxonomy builds on our perception of dynamic access control in IoT contexts

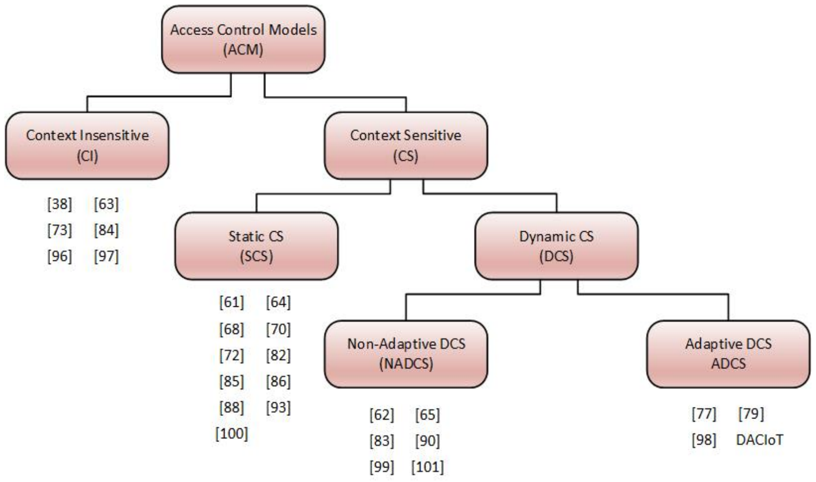


Figure 2.7: Classification Taxonomy for Access Control Approaches in IoT.

We classify access control approaches into two main categories: context-sensitive and context-insensitive approaches. We divide the context-sensitive approaches into static and dynamic, and we divide the dynamic approaches into adaptive and non-adaptive. To explain our classification of access control models and point out the differences between the defined categories, we provide a running example from university domain that shows how access control approaches in different categories make the access decisions and what attributes we consider to group approaches in each category. In Figure 2.7, we assigned 28 papers to our taxonomy by associating the citation of each paper to the category it belongs to.

Context-Insensitive Access Control Approaches: These are the access control approaches that only consider static factors (e.g., identity, role and group membership) in access decision-making and enforcement. For example, in standard RBAC, access permissions are assigned to a roles, and roles are assigned to subjects. A subject entitled with certain role (e.g., student) can only perform operations defined and assigned statically to that role. According to this category, all individuals who are assigned to role "student" can access all university buildings and uses available resources such as Wi-Fi, computers and printers based on the set of permissions statically assigned to this role.

Context-Sensitive Access Control Approaches: These are access control approaches that consider, beside the static factors, dynamic context information (e.g., time, location, relationships, access history) in access decision-making and enforcement. In literature, many approaches can be classified under this category. For example, Generalized Role-Based Access Control (GRBAC) [60] leverages and extends the power of traditional RBAC by incorporating subject roles, object roles, and environment roles into access control decisions. Particularly, environmental roles capture contextual information such as time of day or system load so it can be used to customize the access decisions made based on subject and object roles. According the approaches that fall under this category, all individual assigned to “student” role can access university resources based on permissions assigned to this role. However, these permissions could be restricted by time, location, and historical usage of resources.

Static Context-Sensitive Access Control Approaches: These are the context-sensitive access control approaches in which, at any point of time, the access decision is only dependent on current context information and no historical (e.g., previous authorizations and denials) or prediction information (e.g., size of data to be accessed and usage behavior) is considered. For example, Identity authentication and capability based access control (IACAC) [61] is a context aware access control approach. IACAC uses Identity and context information such as time and location to control access to protected resources. IACAC depends only on current inputs of context information and considers no historical information (e.g., previous access decisions and system status) in evaluating incoming access requests. According to approaches that fall under this category, the students can access a university resource regardless of their previous usage behavior or access level on that resource. For example, if a student’s access logs show non-compliant access behaviors such as multiple denied accesses to school resources they are not authorized to, they can still access their lab resources because only their current context (e.g., time and location) is considered in access decision for lab resources while previous access level (i.e., access to the building) is not.

Dynamic Context-Sensitive Access Control Approaches: These are the context-sensitive access control approaches, in which access decision-making and enforcement are dependent on both current access context as well as historical or prediction information. For example, Capability-based Context Aware Access Control (CCAAC) approach [62] uses context information in access decision-making. In addition to current access context, CCAAC uses historical context information such as user authentication and trust level that could influence future access decisions. According to this category, the access decisions and permissions may differ for two access requests initiated by the same student for the same resource under same current access context, but using different authentication methods and/or the student's trust level has changed in time between the two access requests. For example, if a student uses a fingerprint scan to enter a school building under certain context (e.g., during the weekend), they can have more access permissions (e.g., access to lab rooms) than they could have had if they used an access key to enter the same building under the same context. A dynamic access control system would use the historical information stored in its memory (i.e., authentication method used by the student to access the building) along with student's current context to decide whether to allow the student to access lab rooms or not.

Non-Adaptive Dynamic Access Control Approaches: These are dynamic access control approaches, in which changes in access contexts (current and historical) always have the same effect on future access decisions. In other words, a non-adaptive dynamic access control system can grant an appropriate set of access permissions but does not necessarily satisfy the least-privilege principle (i.e., optimal set of access permissions). These approaches do not adjust their access control policies to improve or optimize the level of authorization. According to this category, students can always use their maximum access privileges, for instance, on WI-FI network regardless of the current threat level or the available network bandwidth. Typically, these approaches lead to unfair access to system resources.

Adaptive Context-Sensitive Access Control Approaches: These are dynamic access control approaches, in which changes in access contexts can have different effects on access decisions. In other words, an adaptive dynamic access control system can go further in their use of context information to satisfy the least-privilege principle. Typically, these approaches define some sort of application-dependent cost function and continuously adjust their access control policies to minimize that cost function. For example, we classify all Risk-based access control approaches under this category. According to approaches that fall under this category, the students could be assigned less access privileges (e.g., download rate) than they are entitled to because, for instance, their previous accesses have led to raising the threat level of the system or caused degradation in system performance or both.

In the following section, we investigate, analyze and classify the access control solutions proposed for IoT from a dynamic access control perspective, and then we will present our findings.

2.7 Literature Survey

Access control schemes have been extensively studied over the past few decades and remain an area of intense research interest due to rapidly changing technology, growing security and privacy concerns, and widespread adoption of new technologies such as IoT. Many efforts were put to extending basic access control models with several features in order to address IoT-specific access control challenges. Research works in this direction can be broadly categorized into proposals to improve the access control policy management, proposals to enable lightweight and real time policy enforcement, proposals to realize policy interoperability in collaborative IoT environments and proposals to establish trust in IoT device-to-device communications.

2.7.1 Improving Access Control Policy Management

Given the large-scale and highly dynamic nature of IoT environments, maintaining up-to-date and conflict-free access control policies becomes a tedious and error-prone task. To simplify policy management, researchers extend standard access control models with several features to improve the flexibility, scalability and adaptability of access control policies to IoT environments. Research works are divided into two approaches: Improving identity management and enhancing policy expressiveness.

- **Improving Identity Management**

  CapBAC model has drawn the researchers' attention due to several unique advantages of the model. In particular, the flexible delegation property makes CapBAC a preferred choice for large-scale networks such as IoT. However, the main issue in CapBAC is the complexity associated with the distribution and revocation of access tokens. In large-scale settings, it is difficult to remember which subject has what permissions on which object. An early work by Gong [38] proposes an Identity-based Capability System (ICAP) approach to simplify the management of access control tokens in distributed network environments. ICAP improves the scalability and efficiency of the CapBAC model. ICAP takes advantage of the delegation property of CapBAC and extends the model by introducing subject identity into the token structure of CapBAC. To keep track of token distribution and revocation history, ICAP maintains an exceptions list and distribution tree, which both are updated every time a capability is created or revoked. ICAP keeps only one copy of an internal capability on the server side for each object, and uses subject identity to differentiate multiple external subject capabilities on the same object. Therefore, ICAP improves scalability by reducing the number of stored capabilities on the server side. However, this work provides no details on the process of access token creation and how access rights or permissions are assigned. In addition, access decisions are solely enforced based on subject identity and no context information is incorporated. Therefore, we classify this approach as context insensitive.

Anggorojati et al. [62] propose a Capability-based Context Aware Access Control delegation model (CCAAC) for federated IoT networks. CCAAC extends ICAP by introducing a new field (C) to the ICAP capability structure, where C contains context information that is related to the delegator, delegate and object to be accessed. The main contribution of this work is to enable IoT devices to delegate access rights to another IoT device based on the attributes and context of the delegate and resources of interest. Therefore, CCAAC improves the flexibility of ICAP by allowing for a wider range of access control rules to be specified dynamically. We classify CCAAC as a dynamic access control model because it uses both current and historical context information (e.g., trust level) in the delegation process. However, CCAAC defines dynamic access control policies based on attributes and context information where a change in trust level always results in the same changes in access privileges assigned to the user. Therefore, we classify CCAAC as non-adaptive dynamic approach.

Despite the flexibility and scalability that CapBAC model shows in delegating access permissions, it inherits the limitation of Identity-based access control models in that it assigns access permissions based on a subject's identity. In CapBAC solutions, researchers focused on flexibility of delegating access permissions and paid little attention to permission assignment and access token generation.

- **Improving the Flexibility of Access Control Policies**

  RBAC is considered the most mature and widely used access control model in organizational settings. This model received a great deal of attention due to its flexibility in managing access control policies. It overcomes the limitations in Identity-based models in that, instead of direct user to permission assignment, it assigns permissions to roles and assigns these roles to users based on their organizational duties and responsibilities. Therefore, adding, removing, and updating access permissions becomes easier and less prone to errors. However, RBAC does not scale to large settings such as IoT. The sheer number of IoT devices and diversity of their access needs can result in a number of roles that equals or exceeds the number of the devices themselves, which complicates the policy management process.

  Jindou et al. [63] propose to integrate Social Network Services (SNS), such as Facebook and Twitter, with RBAC to control access in Web of Thing (WoT) environments. Owners can define policies to control access to their devices based on their profile and social relationships with other users. The authors extend the standard RBAC by considering the user profile and social links in user-role assignment, enabling personalization of access policies. They also consider device attributes (e.g., status, type, functionalities) in role-permission assignments. This allows for more flexible management of the access control on the protected devices because more factors are considered in subject-role and role-permission assignments, and therefore, wider range of access rules can be dynamically specified. While this work supports personalization and flexibility in access policy management, it may not fit highly dynamic environments for the following two reasons: first, it does not consider dynamic attributes such as time and location in access policy specification. For example, an individual may want to allow family members to access his location information only during weekdays and/or while he is in specific locations. According to this access control approach, however, family members can access the location information of other family member anytime and anywhere. And second access policies are manually updated due to the lack of context awareness. For example, a visiting friend can control room lighting or kitchen appliances while in the house, but the owner needs to revoke assigned permissions when the friend leaves. As such, this scheme is categorized as context insensitive.

Zhang et al. [64] propose a service-oriented approach to access IoT resources, where the operations a user can perform on IoT devices are represented as standard services that can be discovered and invoked by a wide spectrum of heterogeneous platforms; achieving seamless integration of IoT physical objects and better device manageability and interoperability. For authorization, the authors extend the RBAC model with context information to control access to IoT devices. In this approach the context information serves two purposes: it constrains the access permissions assigned to users based on their roles, and it addresses the role explosion issue by being able to activate/deactivate roles at runtime. In other words, context information is considered in user-role assignments as well as in role-permission assignments. However, no historical context is used in this work, so we classify this approach as static context sensitive.

Kayes et al. [65] propose a policy framework for context-aware applications called (CAAC). CAAC aims to simplify the management of access control policies in large-scale deployments and highly dynamic systems. CAAC assigns roles to users based on relevant context information and assigns permissions to roles based on purpose-oriented situation information. The relevant context is any information that describes the state of access elements and their relationships. The purpose-oriented situation is the subset of the complete spectrum of access elements that are relevant to a certain goal or the purpose of resource access request. In order to capture dynamic attributes and integrate them in access decisions, CAAC introduces a semantic based policy that uses Web ontology language (OWL) [66] to represent the basic access control elements (e.g., roles, users, resources, and permissions) and Semantic Web Rule Language (SWRL) [67] to reason about dynamic attributes that are not directly obtainable but can be inferred from existing attributes. Although this solution shows promising features for highly dynamic environments, ontology-based solutions are generally characterized as resource-intensive and require optimizations to cater for resource limitation in IoT devices. We classify this solution as non-adaptive dynamic.

ABAC shows superiority over other models in policy specification and management. ABAC uses a wide range of attributes to control access to protected resources. Identity, role, sensitivity, time and location are examples of these attributes. In fact, ABAC can be customized to define and enforce DAC, MAC and RBAC access control policies. However, ABAC is limited by the richness and availability of authentic attributes. In chapter 3 of this thesis, we propose a context-aware automatic access policy specification schema. The main objective of this solution is to prevent non-authorized data access, and to satisfy privacy constraints for authorized access requests in highly dynamic IoT environments. We propose to automate the generation of ABAC policies to overcome the inflexibility in traditional access policy specification techniques, and improve its adaptability to dynamic IoT environments. The solution uses attributes, context and predication to describe the core access control elements. In response to access requests, the schema produces conflict-free access control policies and makes the final access decisions at runtime. Time, location and collocation information is used to describe the contexts under which the core access control element can associate and form an access rule on the fly. The preliminary evaluation in chapter 4 shows that the proposed approach offers greater flexibility and improved scalability in policy specification. We classify our solution as non-adaptive dynamic.

2.7.2 Enabling Real Time Policy Enforcement

While access decision-making process determines the level of authorization, access enforcement is the process that makes access decisions come into effect. Typically, these two processes are implemented separately. In most cases, the decision-making entity is set on high-end machines that perform complex and resource-demanding access control algorithms while enforcement entity is placed closer to the protected resource to enable real time policy enforcement. In IoT settings, enabling real time policy enforcement is challenging due to resource limitations of IoT devices in terms of computation and communication capabilities. Many research efforts were trying to cope with these limitations and realized real time policy enforcement in IoT environments. Aiming at this objective, research efforts follow two approaches, Peer-to-peer and augmented access control.

- **Peer-to-Peer Access Control**

  One approach to enable real time policy enforcement is to implement the access control logic entirely in the end devices enabling them to make and enforce access decisions in Peer-to-peer fashion. Ramos et al. [68] propose a Distributed Capability Based Access Control (DCapBAC) for IoT. The main contribution in this work is that the authorization logic is embedded on the constrained device and does not require a third party to perform authentication and authorization, thus avoiding delays and ensuring trust. This solution is intended to control human access to IoT devices, and represents a promising attempt towards achieving an end-to-end security. For implementation, the authors use technologies that are designed for computing environments (i.e., M2M communication) that have similar requirements as those of IoT. DCapBAC uses the lightweight JSON and CoAP technologies to represent and communicate access tokens. DCapBAC uses public key cryptography infrastructure to realize scalable authentication and authorization. In addition, DCapBAC implements a lightweight and optimized version of the Elliptic Curve Digital Signature Algorithm (ECDSA) [69] at the end device to verify authenticity of access tokens.

Access requests are granted only if their context satisfies the device local conditions. However, DCapBAC externalizes the authorization decisions to a central entity which issues access privileges, in the form of access tokens, to be enforced at the end device.

In a more recent work Ramos et al. [70], extended their work in [68] and present an integrated authentication and authorization approach based on DCapBAC. The approach is developed to control device-to-device communications during the device lifecycle. In their proposed solution, the authors divide the IoT lifecycle into two stages:

– The Bootstrap Stage, during which an accessing device is first authenticated then authorized by obtaining an access token.

– The Operation Mode, during which secure end-to-end communications take place using the credentials obtained in the bootstrap stage.

For the authorization part, the authors use XACML based access control architecture to obtain access tokens, in which permission assignment and decision-making are performed by a non-constrained machine and enforced by lightweight access tokens. In both research works [68, 70], the authors provide no details on the types of context information used in their scenario; hence, these approaches are classified as static context-sensitive.

Mahalle [61], proposes an Identity Authentication and Capability-based Access Control (IACAC) model, which integrates authentication and authorization in one approach for IoT device-to-device communications. The major contributions of this work are: Secret key generation based on Elliptical Curve

Cryptography-Diffie Hellman algorithm (ECCDH) [57], and identity establishment and capability creation for access control. ECCDH is asymmetric key agreement protocol that allows two devices that have no prior knowledge about each other to establish a shared secret key that can be used in any security algorithm. The protocol has received considerable interest for IoT security because it generates keys that provide same cryptographic strength as that provided by RSA [71] keys but with smaller key size and low computation overhead. For authorization, IACAC uses the capability structure of ICAP to control access between end-to-end devices. Access capabilities associate user identity with access rights on objects, and encode access rights as messages to be communicated during the authentication phase to establish access. If the capability that is presented by the subject matches the capability that is stored in the device, access is granted. Although context information is incorporated in this work, we classify IACAC as static context sensitive, primarily because context information is only used in the authentication phase when IoT devices are first commissioned to the network, and not in the authorization process.

- **Augmented Access Control**

  Another approach to real time access policy enforcement is to perform the authentication process at the device level and offload the authorization task to less restricted edge devices (e.g., gateway). Using edge devices brings two advantages in access control context, first it provides the computation resources necessary to adopt standard access control policies and stable technologies; and second it enables for real time access to the context information of the interacting devices and their environment; thus, improves the flexibility of access control policies and ensures real time policy enforcement.

Ning et al. [72] propose an authentication and authorization scheme for the perception layer of IoT. For authentication, the authors use ECC-based keys to establish a lightweight mutual authentication between IoT user and sensor nodes in the perception layer. In this solution, the authorization is offloaded to an attribute management authority that runs on a gateway node. The attribute authority creates and manages user, sensor, and environmental attributes and controls access based on ABAC policy. Attributes are defined as pre-set thresholds in the access policies that should be satisfied by the user's operational context at the access time. This approach combines the benefits of the flexible and scalable ABAC access policies and the real time access to context information. However, since all attributes used in this solution are immutable and no historical context information is considered in the authorization process, we classify this approach as static context sensitive.

Barka et al. [73] propose to adopt RBAC model for access policy specification and use cryptographic keys to enforce the access policies in WoT environments. The authors represent WoTs semantically as ontologies [74] and group them into ambient spaces based on functional similarities (e.g., classrooms). The authors use a customized a version of the ISO reference monitor [75] to build their proposed access control architecture, where two components are responsible for access decision-making: Access Decision Facility (ADF) and Access Enforcement Facility (AEF). ADF intercepts access requests and makes the initial access decision, and then AEF continues to enforce access constraints during the access session. Although this approach supports interoperability in resource representation, it inherits the shortcomings of RBAC in IoT environments in several aspects. For example, this approach has low scalability because the author assumes all devices in the same space have the same access needs, thus one role is defined for the entire space. However, IoT creates scenarios where IoT devices in the same space have different functionalities and have different access needs, which requires the policy administrator to define a distinct role that details the access permission for each device. Further, a new role needs to be manually added/removed every time a device joins/leaves the space since device context is not considered in this approach. In addition, all WoTs share the same private key, which poses a high security threat if this key is compromised as WoTs scale up. Each additional WoT adds another potential point of weakness that an attacker can leverage to break the system. Context information is not considered in this approach either, therefore, we classify this solution as context insensitive.

In chapter 3 of this thesis, we propose a continuous access policy enforcement mechanism for IoT deployments called CPE. CPE describes access control elements using predicates, and stores them as primitive facts in a K-D tree data structure [76]. CPE matches access requests with primitive facts, generates access policies, and makes context-aware access decisions at runtime. In our proposed architecture, the we implement CPE on a gateway node that serves as an authorization server. CPE takes advantage of real time access to contextual information that describes the access element during access session and continuously monitors access control parameters based on which access decisions were made. A performance evaluation is provided in chapter 4 of this thesis to demonstrate the efficiency of the proposed mechanism in highly dynamic environments. We classify this solution as non-adaptive dynamic.

Atlam et al. [77] propose an Adaptive Risk-Based Access Control (AdRBAC) model for IoT environments. This model uses system security risk as the main criterion for making access permissions. The model controls access based on contextual and risk factors including user historical context (i.e., access behavior), resource sensitivity, action severity, and risk history. In their proposed architecture, a risk estimation module uses these contexts with the risk history to estimate the overall access risk value for an access request. Then the estimated risk value is compared with risk policies to determine the access decision. User's access behavior is monitored during the access session and is considered to adjust the system risk value at runtime. The novelty of this approach is that it uses smart contracts to monitor the user access behaviour. The idea behind using smart contract is to facilitate the monitoring process in decentralized IoT settings. Smart contracts are software codes stored in a blockchain [78] and run on a domain local machine. The user behavior is constantly monitored and continuously compared with the smart contract to ensure that the user complies with the terms of the smart contract in order to prevent any potential security breach during the access sessions. Recently the authors extended their work and propose to adopt XACML as policy language and architecture for access control in IoT settings [79]. We classify these approaches as dynamic because they use historical context information in the authorization process, and adaptive because they define a cost function that maps the user access behavior into penalties and adjusts the access control policies based on these penalties at runtime.

An interesting research area that is related but distinct from the ABAC model is the Attribute based encryption (ABE) [80, 81]. ABE is a public key encryption scheme that enables fine-grained access control, scalable key management and flexible data distribution. ABE has two types of implementations:

- Key-Policy Attribute-Based Encryption (KP-ABE): a set of object attributes are embedded in the ciphertext of the encrypted object, and the access policy to the object is embedded in the subject's private key. Only if the access policy in the private key matches that of the encrypted object, then the subject can decrypt the data.
- Ciphertext-Policy Attribute-Based Encryption (CP-ABE): reverses the manner in which KP-ABE works. A user is described by a set of descriptive attributes, and a corresponding private key is issued to the user by an authority. During encryption, an encryptor associates an access policy over attributes with the ciphertext . If and only if the attributes of a user satisfy the access policy of the ciphertext, the user can decrypt the ciphertext.

In the two implementations of ABE, the access policies are embedded either in the subject's key or in the cyphertext, which allows for access policies to be enforced at real time on subject or object side. ABE considers real time context information in access policy enforcement. For example, only recipients who currently live in certain area (i.e., ZIP code) can receive police notifications about certain accidents that took place in that area. Thus, ABE can enforce access policy in real time.

Xinlei [82] provides a performance evaluation study on the usage of ABE in IoT environments. The author aims to evaluate the actual performance of ABE in resource limited computation environments and to understand at what cost ABE offers its benefits, and what situations ABE is best suited for use in IoT. Evaluation metrics include execution time, CPU and memory usage, network overhead and energy consumption in mobile devices. The study concludes that ABE performance output becomes unacceptable with higher security levels, regardless of device computation capabilities, or when used on restricted devices. Thus, without significant improvement, the classic ABE algorithms are best used when the computing device has relatively high computing power and the applications demand low to medium security. In general, we classify access control approaches based on the ABE model as context-sensitive solutions because it considers contextual attributes in access decision-making. However, we classify these approaches as static because once the objects are encrypted and the user's key are established and distributed, the policy trees based on which subject's keys are generated do not change.

Zhang and Wentao [83] propose an access control framework based on UCON model to control access in IoT environments. UCON is a generalization of access control that covers authorizations, obligations, conditions, continuity (ongoing controls), and mutability. The framework leverages the capabilities of UCON model to control access of IoT devices that live in the IoT sensing layer to IoT services that reside in IoT application layer. The main point of this solution is to ensure continuous authorized access during application usage. In their proposed framework, the authors define the service in IoT application layer as a UCON object, the device in the IoT perception layer that is accessing the service as a UCON subject and the reliability of services, trust level of devices as UCON conditions. This framework makes access decisions based on dynamic factors including the trust level and the reliability of the service, and static factors including the device and application attributes. A central trust management authority calculates the trust scores of the devices and continuously updates them according to the feedback provided by the service about the behavior of the device during application usage. The service reliability is assisted based on two properties that are related to the wireless communication space the path loss and propagation distance. For implementation, the authors introduce a fuzzy relationship matrix to assess the trust level between the accessing IoT device and the protected application service. Each element of the matrix represents the level of trust an application service has on a certain device in a wireless sensor network. We classify this approach as dynamic because it considers historical context information (i.e., trust level) in the authorization level. However, we classify this approach as non-adaptive because it maps the final trust values to a set of access permissions based on predefined and static fuzzy rules.

Hussein et al. [84] propose a community-based access control approach for distributed IoT environments. The author borrows the definition of community concept from social life and applies it to IoT context, such that a set of smart objects sharing a common mission and goals forms a community. Based on this structure, less capable smart objects within a community can rely on other objects with sufficient resources to make authorization decisions on behalf of them without referring to a centralized authorization entity. The authors design and implement a proof of concept prototype, where a local authorization server and gatekeeper control access to a set of IoT objects in a smart home. The authorization server is in charge of issuing access tokens for users, and the gatekeeper is responsible for validating these access tokens. Because this approach considers no context information in the authorization process, we classify it as context insensitive.

2.7.3 Realizing Policy Interoperability in Collaborative IoT

In collaborative IoT settings, IoT devices can interacts across various application domains. Each domain defines its own access control policies that meet its security and privacy requirements. Controlling access in such environments require an access control mechanism that accommodates diversity in access control policies and supports seamless decision-making for access requests across collaborating domains.

In capability based access control systems, identity management does not play a critical role, which provides huge advantages especially when managing access control in cross-domain contexts. Gusmeroli et al. [85] present a capability-based access control management system, through which an organization or individuals can control access to their IoT devices within and across administrative domains. The approach supports delegation and revocation of access rights across multiple domains. The authors propose to issue each user an access token that contains reference to the target object, user access rights including delegation of all or a subset of these rights to other users, and token validity time. The token is represented by digitally signed XML files. A valid access token must accompany any access request. To revoke a token, the author propose a delegation chain in the token structure such that a user who owns a capability token can revoke access when needed. The token can be revoked from all descendants at once or from a specific delegate. Because context information is only considered for auditing purposes in this work, and not in the privilege assignment, we classify this approach as static context sensitive.

Liu et al. [86] propose an authentication protocol for federated IoT settings based on OpenID technology [87] and ECC cryptography. OpenID facilitates a single sing on for many different domains by authenticating a single identity provider. The authors provide a formal definition of an authentication protocol and propose a centralized authentication and authorization architecture. They focus their attention on user authentication and key establishment in federated IoT settings. For authorization, the authors propose to adopt RBAC policies; however, no details are given. The only context information used in this approach is the IP address of traffic source and it is not clear how they use this information in the authorization process. Therefore, we consider this approach as static context sensitive.

Savio et al. [88] propose to combine ABAC and CapBAC models to facilitate trusted resource sharing and application reusability across different IoT platforms. The authors emphasis on decoupling the authentication and authorization. During the authentication phase, the application authenticates itself within the home and/or foreign IoT platforms, obtains access token that stores its attributes. Then the access tokens are used by the application during the authorization phase. An application gains access to an IoT resource if its attributes satisfy an ABAC policy defined over that resource. The combination of the ABAC and CapBAC access control models gives this solution the flexibility of ABAC in access policy specification and the scalability of CapBAC in policy enforcement. This solution employs no historical context information in the authorization phase. In addition, it provides no details on the types of attributes they use in the authorization process. Therefore, we classify this approach as static context sensitive.

RBAC access policy has proven great usability and flexibility in organizational settings. However, it does not support access control in federated and collaborative environments. Therefore, researchers have put a considerable research efforts to overcome this limitation. El Kalam et al. [89] propose Organizational Based Access Control Model (OrBAC). OrBAC is an extension of RBAC and is proposed to control access to resources in organization premises. OrBAC introduces the *"Organization"* concept as a new dimension to the RBAC basic model. The main goal of OrBAC is to improve flexibility in policy specification by enabling policy administrators to define access control policies using abstract entities only. Therefore, the policy specification is completely separated from its implementation. OrBAC abstracts subjects into roles, actions into activities, and objects into views. Thus, OrBAC policy specification is completely parameterized by the organization so that it is possible to handle several access control policies associated with different departments in same organization. OrBAC is not restricted to permissions, but also includes the possibility to specify prohibitions and obligations. In addition, OrBAC considers different types of context information (e.g. temporal, historical and spatial contexts) in controlling accesses to organization assets. However, OrBAC does not support access control across different security domains (i.e., organizations) because the scope and meaning of access policy parameters (e.g., roles) may vary across these domains, which raises the need for these parameters to be agreed on before collaboration can take place. Pasquier et al. [90] propose SmartOrBAC to enhance the manageability and flexibility of the OrBAC access control policies in IoT collaborative environments. SmartOrBAC introduce a collaboration layers within a centralized-distributed authorization architecture. The main function of this layer is to define access control policies for different organizations, in an attempt to address the collaborative aspect of IoT environments, for which the OrBAC model falls short. In addition, the proposed access control architecture allows for real time access to context information, which enables SmartOrBAC to define more context-aware access control policies. However, SmartOrBAC inherits OrBAC's shortcomings in terms of the manner in which context information is used in policy specification. Both models consider context information only in role-permissions rather than user-role assignment. In addition, changes in organization structure (e.g., adding new role) will trigger reconfiguration of the whole system, rendering OrBAC and its extensions [91, 92] unfit in highly dynamic environments such as IoT. We classify this approach as dynamic because it maintains a short memory for preceding steps such as authentication and trust level in the authorization process. However, we deem this approach non-adaptive because it always grants the same access permissions under the same access context, and provides no techniques to improve the level of access privileges assigned in certain access context.

2.7.4 Establishing Trust in IoT Environments

Data privacy, integrity, and availability are the major ingredients of trust in an IoT context. IoT devices transfer sensitive data that require high levels of privacy. Therefore, attention should be paid to the contexts under which the data is produced to be able to conclude whether it introduces any privacy issues. Data integrity is no less important in IoT ecosystem; corrupting the data on which an IoT device relies could make device dangerous to use or exploited to become a compromised platform from which other attacks can be launched. Availability is equally important as privacy and integrity. The availability requirements depend on the IoT application. While lack of availability will not cause dire consequences in some IoT applications (e.g., weather forecasting), it might be critical in others (e.g., health care). Therefore, IoT entities need to negotiate and exchange some form of credentials before each entity can trust the other and share knowledge between them. However, establishing trust in highly dynamic IoT environment introduces significant challenges to manage identity, authentication and authorization.

To ensure privacy-aware access to IoT user's data, Luis et al. [93] presents an access control mechanism for IoT environments by modelling IoT communications as resources, and uses User Managed Access (UMA) [94] to protect these resources. UMA is an OAuth-based access management protocol that enables a resource owner to control access to their data and other protected resources. UMA provides the resource owners a unified control point of authorizing who and what can have access to their data, content, and services, no matter where all those resources are stored. The authors integrate UMA into the Message Queuing Telemetry Transport (MQTT) communication protocol [95] to provide a unified access control scheme that abstracts heterogeneity of IoT devices and protects the communication of data between IoT devices. We classify this approach as static context sensitive because it makes access decisions based on only user identity and device IP address, not on historical context.

Al ALkeem et al. [96] integrate RBAC and ECC to preserve data privacy and to minimize potential security attacks to wearable devices in healthcare. The authors pointed out that security attacks may take place at three phases: data collection by wearable devices (e.g., sugar level sensors), communication via intermediate devices (e.g., smartphones) and data processing at a remote server (e.g., patient's records database). Their solution controls access to patient records stored at the cloud by classifying these records in advance into different sensitivity levels (secret, restricted and public) and map them to static and predefined roles and permissions. A combination of a PIN and biometric tokens is used to augment authentication in this framework, while ECC is used for key management. Once authenticated, users who have designated roles (e.g., doctor) can access patient' records based on permissions assigned to each role. The authorization is performed at the remote server with no consideration to any context information. Therefore, we classify this solution as context insensitive.

Dorri et al. [97] study access control issues in smart home scenarios. The focus of this work is to leverage the blockchain technology to provide a distributed and trustworthy access control scheme. In the proposed architecture, there is a local resource manager located in every home that handles devices and their communications called miner. Each home miner maintains a local private blockchain with a policy header storing access control policies to control all incoming and outgoing requests. The policy header has four parameters: (1) the requester's received public key, (2) requested actions (which include store, store cloud, access and monitor), (3) device ID, and (4) action to be performed should all submitted parameters match existing policies. The solution, however, does not use context information in creating and consuming the access transactions. Therefore, according to our classification, it belongs to context insensitive category. Yuanyu et al. [98] propose a smart contract-based access control framework for IoT systems. Similar to the work done in [97], the framework uses Blockchain to provide a distributed and trustworthy access control scheme. The framework implements three types of smart contracts: Access Control Contract (ACC), the Judge Contract (JC) and Register Contract (RC). The main idea is to use these contracts to implement access control lists (ACL), where each row in the list corresponds to one subject-object pair and provides an access rule defined on a certain (resource, action) pair. The framework supports two methods for access rights validation: a static method based on predefined access control policies and dynamic method based on usage behavior. JC receives misbehavior reports from ACCs and returns a corresponding penalty to the ACC. The ACC then takes countermeasures based on the penalty decision. We classify this approach as dynamic and adaptive.

Junshe et al. [99] propose a trust and attribute-based access control for IoT. The authors design a trust attribute-based access control model, which integrates a dynamic trust attribute and multiple static attributes. The trust score is calculated based on the set of trust evidence provided by the user such as authentication information and current context. According to the user trust evidence, the trust evaluation module adjusts the trust level of the user and updates the trust attributes database. We classify this approach as dynamic as it uses historical trust average value in the authorization process and non-adaptive because it maps the final trust values to a set of access permissions based on predefined and static fuzzy sets.

Ouaddah et al. [100] propose and implement a distributed access control framework called FairAccess. The framework leverages the Blockchain technology to control access to IoT resources. Unlike financial bitcoin transactions, the authors introduce a new type of transaction to grant, revoke and delegate accesses. An access token is generated and encrypted by resource owners using a public key cryptography. This token is broadcasted as one transaction to the network. The transaction includes the attributes, context and private key a potential requester needs to fulfill in order to accept this transaction and gain access to the protected resource. All transactions are stored in the blockchain and verified by other nodes in the network. We classify this approach as static context sensitive, because it only uses current access context to validate access tokens and does not consider historical context information in the authorization process.

Parikshit et al. [101] propose a fuzzy approach for trust based access control called FBAC. This approach controls access to IoT devices based on the trust level a device has on other devices. Trust score is calculated based on three factors including the knowledge a device has about other devices, experience of previous accesses, and recommendations. Depending on the resulted trust value, trustworthiness of the other device is decided and used for permission mapping to achieve access control. We classify this approach as dynamic, because it considers historical information such as previous interactions, and non-adaptive because it relies on a predefined and static fuzzy set to map trust scores to access permissions.

2.8 Evaluation of IoT Oriented Access Control Solutions

We performed a qualitative evaluation of related literature based on the major access control requirements in IoT and our dynamic access control taxonomy. Evaluation attributes include scalability, distribution, interoperability, real time enforcement, lightness, context awareness, dynamic, and adaptive.

Table 2.7: Evaluation of Access Control Approaches Proposed for IoT

Citation	Solution Name	Basic Model	Scalability	Distribution	Interoperability	lightweight	Real time Enforcement	Context-Sensitive	Dynamic	Adaptive
[61]	IACAC	CapBAC	Yes	Yes	Yes	Yes	Yes	Yes	No	No
[62]	CCAAC	CapBAC	Yes	Yes	Yes	NC	No	Yes	Yes	No
[38]	ICAP	CapBAC	Yes	Yes	Yes	NC	No	No	No	No
[63]	Social network-Based	RBAC	Yes	No	Yes	NC	NC	No	No	No
[64]	Service-oriented	RBAC	No	No	Yes	NC	Yes	Yes	No	No
[65]	CAAC	RBAC	No	No	No	No	Yes	Yes	Yes	No
[68]	DCapBAC	CapBAC	Yes	Yes	Yes	Yes	Yes	Yes	No	No
[70]	DCapBAC-Ext	ABAC-CapBAC	Yes	Yes	Yes	Yes	Yes	Yes	No	No
[72]	No Name	ABAC	Yes	No	Yes	Yes	Yes	Yes	No	No
[73]	Ontology-Based	RBAC	No	No	No	No	Yes	No	No	No
[77]	AdRBAC	RBAC	Yes	No	NC	NC	Yes	Yes	Yes	Yes
[79]	AdRBAC-Ext	RBAC	Yes	No	Yes	No	Yes	Yes	Yes	Yes
[82]	ABE	ABAC	Yes	No	Yes	No	Yes	Yes	No	No
[83]	No Name	UCON	Yes	No	Yes	No	Yes	Yes	Yes	No
[84]	Community-driven	NC	Yes	No	NC	No	Yes	No	No	No
[85]	No Name	CapBAC	Yes	Yes	Yes	Yes	Yes	Yes	No	No
[86]	No Name	RBAC	Yes	No	Yes	Yes	No	Yes	No	No
[88]	No Name	RBAC-CapBAC	Yes	No	Yes	No	No	Yes	No	No
[90]	SmartOrBAC	RBAC	No	No	Yes	No	No	Yes	Yes	No
[93]	No Name	UMA-OAth	Yes	Yes	Yes	Yes	Yes	Yes	No	No
[96]	No Name	RBAC	No	No	No	Yes	No	No	No	No
[97]	BlockChain-Based	ABAC	No	Yes	No	No	No	No	No	No
[98]	No Name	ACL	No	Yes	Yes	No	Yes	Yes	Yes	Yes
[99]	No Name	ABAC	Yes	No	Yes	No	Yes	Yes	Yes	No
[100]	FairAccess	Other	No	Yes	No	No	Yes	Yes	No	No
[101]	FBAC	NC	Yes	Yes	No	Yes	Yes	Yes	Yes	No
-	DACIoT	ABAC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

In chapter 3, we propose a Dynamic Access Control Framework for IoT (DACIoT). The framework supports three dynamic access control mechanisms that we deem lacking in existing access control solutions: (1) Automatic policy specification, (2) Continuous policy enforcement and (3) Adaptive policy adjustment. In our framework, we use historical context information in access decision-making. We also use current context information to generate and adjust attribute-based access control policies at runtime. Therefore, we classify our work as dynamic and adaptive.

We claim that our approach provides higher security than other approaches proposed for IoT because it provides better adaptability to frequent changes in security and privacy requirements in highly dynamic IoT environments. Unlike traditional access control policy specification methods, our approach generates conflict-free access rules at run time based on always up-to-date primitive facts eliminating the need for resource-demanding rule conflicts detection and resolution algorithms. In addition, our approach continuously applies the principle of least-privileges such that IoT users/devices always have the minimum access permissions they need to perform a task. The principle is preserved through constant monitoring of changes in access contexts. To the best of our knowledge, our approach is the first to address changes in access policies while IoT devices are in use. Our approach adjusts access policies based on changes in device access behaviors to prevent excessively privileged users in control of these device from misusing system resources.

Table 2.7 summarizes the results of our study. For each attribute, we indicate with a "Yes" and "No" whether or not the access control solution supports the corresponding criteria, and the Not Clear abbreviation "NC" to indicate that the research presented insufficient information with regard to the attribute of interest. We also provide the citation, solution name (if any) and basic model.

2.9 Open Research Issues

Based on our analysis and evaluation of the current literature, we identify the following research gaps in access control research:

1. Existing access control solutions are proposed to protect data that is permanently stored, and rely on manually configured access control policies that are rarely to change at rune-time. Typically, the management and maintenance of these policies are tasked to experts policy administrators. In addition, these

solutions assume the existence of complex mechanisms to detect and resolve potential conflicts in access policies and rules. However, the dynamic nature of IoT environments, in terms of the number of users, resource availability, spontaneous and opportunistic interactions among IoT devices, creates access scenarios in which an authorized access becomes prohibited and vice versa at run-time, which raises the need to frequently update the underlying access control policies on the fly. In addition, IoT users might not possess the necessary policy management skills to define and maintain access control policies that satisfy their security and privacy requirements, not to mention that existing rule conflict detection and resolution algorithms are resource-intensive, which contradicts the limitation of computation capabilities of IoT devices. For these reasons, it becomes a necessity to automate the process of policy specification to simplify the generation and maintenance of conflict-free access control policies, overcome the inflexibility in traditional access policy specification techniques.

2. Existing context-aware access control solutions verify the context information only at the time the access request is evaluated, and then assigns access privileges accordingly. It does not consider changes in context after the access is granted. Such an approach can expose system resources to unauthorized access if the context of the access changes during and access session. However, advancements and convergence in IoT enabling technologies along with ubiquitous connectivity have led to the generation of new wave of smart services and applications based on real-time data access. The popularity of ubiquitous data access and accelerated adoption of these services pose significant challenges on user and data privacy. Thus, controlling access to such services in highly

dynamic environments with continuously changing context becomes even more challenging. A delay in making access decisions when context changes may result in consequences that cause harm and property damage. Therefore, continuity in access policy enforcement becomes a necessity in highly dynamic IoT environments for the entire access session not only at the time of request.

3. Despite the considerable research efforts that are put to improve the flexibility of access control policies, these policies can still assign more access privileges than that actually needed by requesting entities, which exposes systems resources to insider attacks. The risk of insider attack is further exacerbated in IoT environments because access control policies become obsolete quickly due to frequent changes in the access contexts under which the IoT devices can be accessed. In addition, the insider attack can exist in IoT environments and continue for a long period of time without being noticed or treated. Detecting insider attacks in such environments is quite challenging and presses the need for an adaptive access control mechanism that can react instantly to abnormal access behaviors and adjust the access control policies at runtime with minimal or no human intervention.

2.10 Summary

This chapter describes and evaluates the state-of-the-art access control solutions proposed for IoT, pointing out enabling technologies and bringing forward various challenges and open research issues. The IoT computing paradigm has emerged due to the advancements in communication and computation technologies. IoT opens up a new set of smart applications and brings many advantages that led to better quality of life; ubiquitous connectivity makes more information available for better decision-making, automation and control saves time, money and maintain quality of service, monitoring provides information that could not have previously been collected easily which provides an advantage of knowing things in advance and taking the necessary actions. Many access control solutions are proposed in literature to reduce the security and privacy concerns about unauthorized access to data generated, communicated and processed in IoT ecosystems. The proposed access control solutions extend one or more of basic access control models with several features to cater for IoT specific access control requirements. ABAC has proven to be a natural choice for large-scale and open IoT environments. The chief advantage of ABAC model is that it controls access based on a set of attributes that describe the requester, resource and environment, which allows for definition of flexible access control rules based on a wide range of attributes rather than identity, role or security levels. Regardless of the adopted access control model, existing access control solutions rely on static and predefined access control policies. However, IoT environments are highly dynamic and the context under which the IoT devices can be accessed constantly changes, thus the sensitivity of device's resources, which raises the need to continuously update the underlying access control policies. Efficient access control mechanisms that are capable of capturing the dynamic nature and accommodate the frequent changes in security and privacy requirements of IoT environments are definitely crucial to the widespread adoption of IoT technology.

In conclusion, although significant steps have been taken towards controlling access in highly dynamic IoT environments, many research challenges remain open and need further investigation.

Chapter 3

Dynamic Access Control Framework for IoT

3.1 Introduction

Given the unique access control challenges that IoT poses, researchers have focused on optimizing specific aspects of current access control policies, models and mechanisms such as improving access control policy management [63, 64] or enabling real-time policy enforcement [68, 69] to cater for the dynamic nature of IoT environments. However, the lack of a comprehensive understanding of context-aware access control, and the various constraints of IoT devices, render most of these approaches incapable of efficient and reliable access control in IoT scenarios. This chapter introduces a Dynamic Access Control Framework for the Internet of Things (DACIoT). DACIoT supports three functionalities that are lacking in existing access control solutions (1) Automatic policy Specification, (2) Continuous policy enforcement and (3) Adaptive policy adjustment. DACIoT integrates these functionalities in a holistic and context-aware access control framework that addresses various aspects of efficient access control in highly dynamic IoT environments. The main objective of DACIoT is to prevent unauthorized access to IoT devices and tightens the authorized access while an IoT device is in use. DACIoT integrates access context in the access policy specification, enforcement and adjustment, to overcome the inflexibility in traditional access policy specification and maintenance techniques, ensure continuous and up-to-date policy enforcement, and improve the adaptability of access control policies to dynamic IoT environments. This chapter

- Introduces DACIoT as extension to the successful eXtensible Access Control Markup Language (XACML) reference model defined by the Organization for the Advancement of Structured Information Standards (OASIS) [51].
- Introduces the concept of automatic access policy specification and describes the algorithm underlying the run-time generation of access rules.
- Introduces the concept of continuous access policy enforcement and describes the developed algorithm for run-time re-evaluation of ongoing access sessions.
- Introduces the concept of adaptive access policy adjustment and describes the algorithm that runs the the access behavior classification and access policy adjustment processes.
- Presents a decentralized proxy-based access control architecture for DACIoT, and describes DACIoT workflow.

The reminder of this chapter is organised as follows. Section 3.2 presents XACML standard reference model. Section 3.3 describes the proposed access control framework and relevant research efforts that may be potentially applied for each component. Section 3.4 presents the access control architecture and workflow of our framework. Section 3.5 draws the concluding remarks of the chapter.

3.2 XACML Reference Model

XACML provides a common ground for the access control terminology and implementation [102, 103]. The standard provides access control architecture, policy language and a request/response protocol. XACML uses Attribute-Based Access Control (ABAC) policies to control access based on a wide range of attributes including subject , resource, and environment attributes. This allows XACML to describe general access control requirements by combining multiple access control policies, as well as be customized to support specific business access control needs, for example, XACML can be specialized to implement RBAC policies, where the role is considered the main attribute that determines the authorization level of a subject on a certain resource. In the following, we describe the XACML access control components, their functionalities and interactions.

- Policy Enforcement Point (PEP): PEP intercepts access requests to system resources and forwards them to the Policy Decision Point (PDP) where access decision is made. PEP executes the access decisions it receives from PDP. It implements fulfilling obligations that PDP may include in the access decisions. An XACML obligation is an instruction from the PDP to the PEP on what action(s) must be performed before and/or after an access is approved.

- Policy Decision Point (PDP): PDP evaluates access requests and makes access decisions based on available attributes that describe the access requests elements (subject, object, operation), and applicable access control policies.

- Policy Information Point (PIP): PIP acts as the source of attributes that PDP requires to evaluate access requests against ABAC policies. PIP collects attributes that pertain to the subject, resource of interest and environment in which the access requests take place, and provide these attributes through the context handler to the PDP upon access decision-making.

- Policy Administration Point (PAP): PAP supports policy management functionalities including adding/removing and modifying access policies. It also stores the access policies defined by the policy administrator.
- Context Handler (CH): controls the workflow of the access control system. It represents a hub through which PEP, PDP, and PIP communicate. CH forwards access requests it receives from PEP to PDP and returns the access decisions from PDP to PEP. It also fetches the attributes and resource content required for PDP to make access decisions.

It is worth mentioning that XACML organizes access policies into three levels including: policy set, policy, and rule. XACML uses targets to index policy sets, policies and rules. A target is a predicate of one or more variables defined on the subject, the resource and the operation of access requests; it specifies the type of requests a policy set or policy can be applied to. If a policy applies to a given access request, its rules are evaluated to make the access decision, otherwise, the policy is skipped.

3.3 DACIoT: Dynamic Access Control for IoT

Figure 3.1 depicts the XACML reference model components extended with our framework components (shown in dashed lines). In our proposed model we replace the three

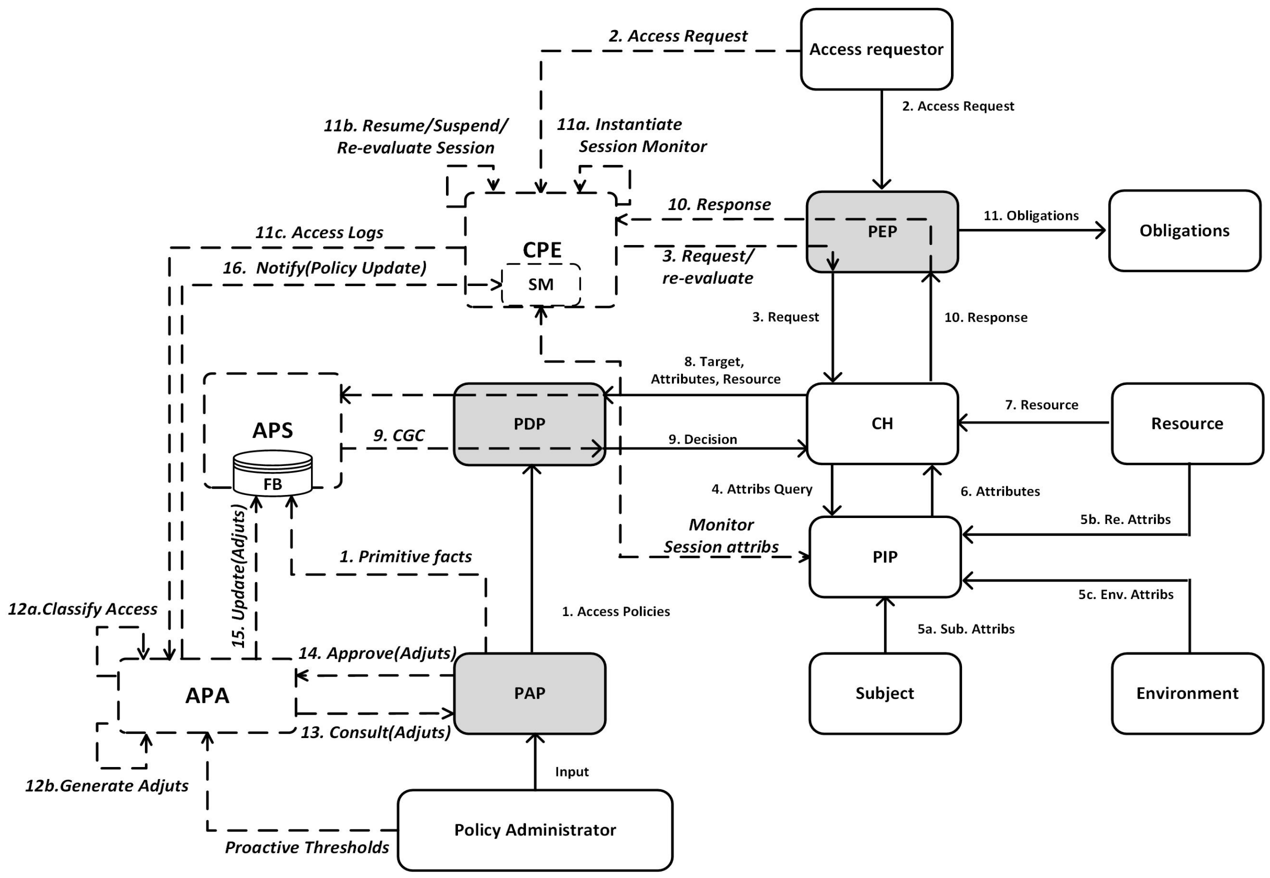


Figure 3.1: Extended XACML Reference Model.

(shaded) components of XACML: PDP, PEP and PAP with an automatic policy specification schema (APS), a continuous policy enforcement mechanism (CPE) and an adaptive policy adjustment technique (APA) respectively. To follow the access control standards, we define the functionalities of each component of our framework within the scope of its XACML counterpart. The main functionalities of the APS are to evaluate access requests provided by the CPE, generates access rules at runtime, and makes final access decisions. The CPE executes the access decisions that are made by APS and ensures that the access rules, based on which access was granted, are always satisfied while a resource is in use. The APA adjusts the access rules based on potential changes in device access behaviors such that accessing devices always have an appropriate level of access permissions. The difference between the APA and CPE components is that the APA can change or modify the access rules by restricting or relaxing certain conditions in these rules, while the task of the CPE is to execute the modifications by triggering a re-evaluation process of ongoing access sessions against these modifications.

In the following, we describe the functionalities of DACIoT major components.

3.3.1 Automatic Policy Specification

One limitation of traditional ABAC access control policy is that it describes core access control elements and their attributes holistically. Holistic in the sense that policy administrators need to consider all possible associations of the access control elements and define the attributes and conditions that govern every single association (i.e., rule) in the policy specification process. The tight coupling describes parallel relationships among access control elements, which limits the flexibility in policy expression. For example, if the policy administrator is to add new type of subject to the access policy, they need to define an access rule that describes the attributes and conditions that associate the subject with every type of object he is authorized to access. The same applies for adding new types of object or operation.

Another limitation of ABAC policy is that it is static in the sense that access policies are specified at the setup time and do not change. For example, if the policy administrator is to define an access rule that increases or limits the access permissions of a certain subject on a certain object, the policy administrator needs to look up and update all predefined access rules that associate with the subject and object. Otherwise, the new access rule would create conflicts granting the subject inconsistent access permissions on the same object. Such an approach limits the adaptability of access control policies in dynamic access scenarios; it complicates the policy management and introduces significant policy maintenance overhead (e.g., policy conflict resolution).

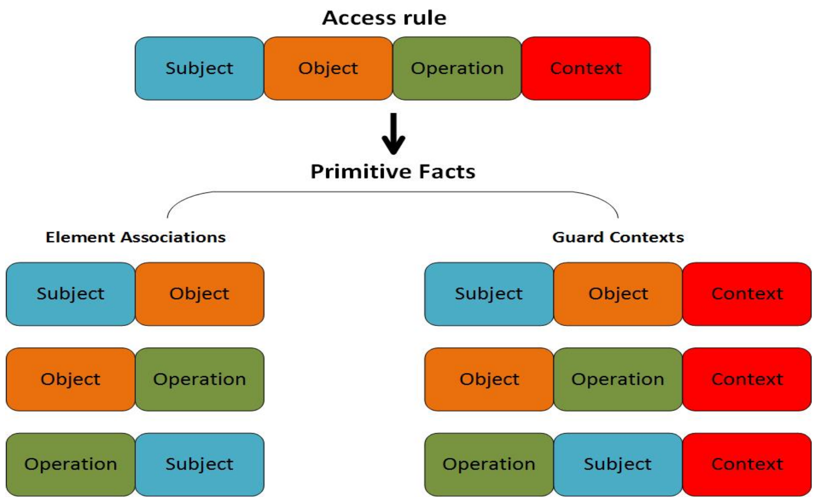


Figure 3.2: Primitive Facts.

To overcome the inflexibility in policy specification methods, we introduce an Automatic Policy Specification (APS) schema. APS breaks down access control rules to their fundamental elements: subject, operation, and object, and creates two types of association: element-element and element-element-context. The former, which we call the element association, simply associates elements into subject-object, object-operation, operation-subject pairs. The latter, which we call the guard context, attaches the contextual conditions described in the original access rule to all element associations. Unlike conventional policy specification methods, we do not store access rules in their final format, rather we store the element associations and their guard contexts as primitive facts in the policy database as shown in Figure 3.2. When an access request is received, an access rule(s) is automatically generated if there exists in the policy database element associations that match the elements of the access request. The access is granted if the real-world conditions of the request elements satisfy the guard contexts defined for each element association. Otherwise, access is denied by default.

Guard Context (GC) is a set of predefined application dependent values or thresholds represented as key-value pairs for example (location: "lab room"), (time, "9:00 am"). These key-value pairs describe the environmental conditions defined by the object administrator under which a subject of certain attributes (e.g., identity, role, and access credits) can perform operations of certain attributes such as (type: "read"), (granularity: "per minute") on an object of certain attributes such as (CPU utilization: "$< 70\%$"), (energy level: "$> 80\%$").U utilization: "$< 70\%$"), (energy level: "$> 80\%$").

Operational Context (OC) is represented by key-value pairs similar to GC. However, values in OC are real-time measurements that reflect the real-world conditions of the subject, object, and the requested operation at the time of access. If these measurements satisfy the GC conditions that are defined separately for each element, then access is permitted. Otherwise, access is denied by default.

Primitive Facts (PF) describe the core access control elements using abstractions, in a key-value pair representation, which contain both the attributes that characterize elements and the guard context that determines the qualification context (or constraints) relevant to each element that controls access to objects. We build these basic abstractions and represent them in predicates as follows:

- Element Abstraction

element (X) is a descriptor represented by a tuple of the form {`type:value, key1: value1, key2: value2, ...`}. The tuple contains a key *type* whose value is {`subject|operation|object`} to indicate this tuple is pertaining to the which of the access core elements. A descriptor is likely to contain a *time* and *location* keys that identify a certain time frame and location that control access to objects. For example, a subject must be in location x and time t to perform an operation of a certain object. The element descriptor is a unified fact defined by the object administrator on attributes of the access elements and their associated context. For example, the following basic abstraction represents a factual tuple for a subject x:

element x= {`type:"subject",name:"Any",study-level:"undergraduate", advisee:"True",location:"IoTResearchLab",time:"6:00-16:00"`}.

This descriptor contains a type key, three subject attributes that basically identify the subject, and two access constraints that determine the guard context required to gain access to a protected object.

- Request Abstraction

  The request abstraction consists of a request template that defines the request elements and OC. The request descriptor contains patterns of the form:

  p= {`type:value, key1:value1, key2:value2, ...`}.

  For example, the following tuple represents a print service request (pertaining only to operation):

  p={`type:"operation", name:"print", location:"floor 6", time:"8:00-14:00"`}.

A request descriptor matches an object if there is an operation in the fact base that satisfies every pattern in the request template.

In listing 3.1, the access policy specification algorithm takes the primitive facts and the access request parameters as inputs and automatically produces the access control decision and the *Common Guard Context (CGC)* as an output. When APS receives

Listing 3.1: Automatic Access Policy Specification Algorithm

```
input  : Primitive Facts $PF$, Access Request $AR$
output: Access Decision, CGC
1  begin
2      for $AR$ do
3          Extract all attributes values of the subject
4          $x = \{type : “subject”, key_1 : value_1, key_2 : value_2..\}$
5          Extract all attributes values of the operation
6          $y = \{type : “operation”, key_1 : value_1, key_2 : value_2..\}$
7          Extract all attributes values of the object
8          $z = \{type : “object”, key_1 : value_1, key_2 : value_2..\}$
9      end
10     Generate $PF\text{-}query\{x\}$, $PF\text{-}query\{y\}$, $PF\text{-}query\{z\}$.
11     Result := Deny
12     for all element $\in PF$ do
13         if $\exists$ subject(x) and operation(y) and object(z) in PF: then
14             Result:= Permit
15             CGC:= $GC_{subject} \cap GC_{operation} \cap GC_{object}$
16             Break
17         end
18     end
19     Return (Result,CGC)
20 end
```

an access request (i.e., target and attributes) from the context handler, it extracts the attributes associated with the access request at runtime, lines 2 to 9, and uses the primitive facts, stored in the Fact Base (FB), to dynamically compute access rules and make an access decision in response to the access request at hand, lines 10 to 18. In line 15, the algorithm calculates the CGC, which is the intersection of GCs associated with the subject, operation, and object elements of the generated access rule. CGC is a core parameter upon which *continuous policy enforcement* is ensured during an access session.

Breaking the access rules down into primitive facts has many benefits: (1) it enables the access control system to generate and update access rules automatically and at runtime, (2) it allows the access control system to reuse primitive facts in the generation of access rules, thus reducing the space complexity as the number of rules increases, (3) it removes the burden of detection and resolution of access rule conflicts because access rules are stored in form of primitive facts and a change in one fact would automatically effect all possible access rules that can be generated based on this fact.

3.3.2 Continuous Policy Enforcement

The literature of access control models that use context information in making access decisions divide into two implementation approaches. The first (and most simply implemented) is discontinuous enforcement, this approach verifies the context information, assigns the access privileges and makes the access decision only at the time access is requested. It does not consider changes in context after the access decision is made such as the work by authors in [38, 60, 61, 62]. This approach can result in security and privacy breaches by disclosing system resources to users whose access context has changed since the time the access decision was enforced, and therefore they become non-authorized users.

The second is continuous enforcement. This approach constantly monitors the operational context of an access session and continuously updates the assigned access privileges according to changes in access context and discloses system resources only to those users who are authorized under the current context. However, this approach, if not carefully implemented, may place considerable computation and communication burdens on the access control system. While discussion on the first approach has dominated the research area in recent years, there is little, if any, research investigating the practicality of the second approach in high dynamic environments such as IoT.

In our framework, we introduce a Continuous Policy Enforcement (CPE). At the core of our proposed design and implementation of CPE component, we aim to capture changes in access context at fine granularity and continuously enforce the appropriate access permissions in highly dynamic IoT scenarios while not posing significant computation or communication overhead. CPE can handle two types of context changes: when the OC changes, which are more frequent and happen as a result of changes in real-world conditions of the access elements (e.g., mobility); and when the GC changes, which are less frequent and happen as a result of updating the access policies (i.e., add/remove access rules), or inferring new facts that could affect the access decision of an ongoing access session. The CPE replaces the PEP component of XACML and supports the following functionalities:

1. **Session Registry**

   The Session Registry (SR) stores a status record for each active session. The status record maintains the session id, the initial GC upon which the request was granted, and a status flag that indicates the current status of an access session (e.g., active, inactive or suspended).

2. **Session Monitor**

We extend the CH with a Session Monitor (SM) functionality that continuously reads the environmental attributes from available sensors, and potential policy update notifications from the *Adaptive policy adjustment component*. The SM keeps track of changes in the OC and CGC associated to the access elements involved in an active session.

Listing 3.2: Continuous Access Policy Enforcement Algorithm

```
input     : Target T , CGC, Session registry $SR$,
            $policy\_update$
output    : Session History $SH$
initialize: $SR \leftarrow \{\}$ , $SH \leftarrow \{\}$
begin
    if $T \notin SR$ then
        Create new session S
        $S$.id := T // Attributes
        $S$.status := Active
        $S$.OPC := new SM ( T ) // monitor
        $SR \leftarrow \{S\}$
    end
    // Update CGC of re-evaluated sessions
    $S$.CGC := CGC
    while $True$ do
        if $policy\_update = False$ then
            for $all\ S \in SR$ do // Check OPC changes
                if $S.OPC \nsubseteq S.CGC$ then
                    $SR$.Remove($S$)
                    $S$.terminate()
                    $SH \leftarrow \{Req_{T,CGC} terminated\}$ // Logs
                end
            end
        end
        else
            for $all\ S \in SR$ do // re-evaluate sessions
                CH ( $S$.id )
                $policy\_update = False$
            end
        end
    end
end
```

When APS grants a new access session, it passes the target, access decision and CGC of the granted access request to the CPE component. In Listing 3.2, CPE uses the information it receives from APS and performs the following steps:

1. It creates an access session that connects the subject to the resource and stores the session parameters in the SR, lines 4-9.

2. It initiates an instance of SM to monitor the OC of the access session, line 6.

3. In lines 12-18, the SM enforces the CGC constraints over the lifetime of an access session; it ensures that the initial CGC of an ongoing access session is always satisfied by the current sensor inputs. Otherwise, the access session is terminated. If the Primitive Facts (PF) are updated, our algorithm, lines 21-24, does not interrupt the ongoing access sessions, rather it re-evaluates all ongoing access sessions against the changes in the PF. For each ongoing access session a re-evaluation access request is submitted to the CH, and the CH proceeds with normal request evaluation procedure. If an access request is denied upon re-evaluation, our algorithm, line 9, updates its CGC with the new value calculated by APS (ϕ in case of denial), therefore, the SM terminates the corresponding session once it checks the session OC against the new CGC. Otherwise, the access session resumes, however, with a new CGC that may restrict, relax or keep the access permissions previously associated with the access session.

4. CPE provides the access logs it collects in line 16 to the adaptive policy adjustment component for further processing.

3.3.3 Adaptive Policy Adjustment

A major limitation in existing access control approaches is that they rely on policy administrators to define access control policies that always assign the appropriate access privileges to requesting entities. These approaches assume that all access conditions under which system resources can be accessed are known beforehand and are unlikely to change. However, the highly dynamic nature of IoT environments creates access contexts in which pre-defined access control policies can not meet the security and privacy objectives of the policy administrator. In these access contexts, obsolete access control policies may either assign less access privileges than that actually needed by requesting entities; blocking access requests that become legitimate (e.g., emergencies), or assign more access privileges than that actually needed by requesting entity; exposing system resources to insider attacks [104]. While the former case reduces the availability and utilization of system resources, the latter case can extend further to compromise the confidentiality and integrity of these resources. In our framework, we give priority to addressing the problem of over-provisioning of access privileges and leave the former case for future work.

In IoT context, insider threats come from current or former connected IoT devices which have or had access privileges on the system resources, and the users in control of these devices, intentionally abuse these access privileges in a manner that negatively affects the security and privacy objectives of the resource's administrator. Unlike traditional internet devices, a compromised IoT device can cause damage that extends to the physical world. The severity of these damages increases in sensitive contexts especially when these devices are controlled by system insiders. Detecting abnormal access behaviors such as insider attacks in highly dynamic IoT environments is challenging due to highly dynamic access contexts under which the IoT device can be accessed. In addition, access control policies become obsolete quickly in IoT environments due to frequent changes in security and privacy requirements, which further increases the attack surface of the system resources.

Recently, detection and prevention of insider threats has attracted the interest of researchers in the IoT security field. A number of solutions have been proposed to approach insider threats in IoT environments based on behavioral models and anomaly detection techniques [105, 106]. Although these approaches do not focus on the adaptation of the access control policies as a countermeasure to prevent insider attacks, they give great insights for insider threat detection in IoT environments.

In our framework, we are introducing an Adaptive Policy adjustment (APA) technique that reacts instantly to changes in access context and adjusts the access control policies at runtime with minimal or no human intervention. With this, we prevent users of IoT devices from abusing their access privileges or exploiting obsolete access control policies to gain unauthorized access. We leverage the experience from the field of anomaly detection to build the APR components. APR classifies the device access behaviors based on proactive measures provided by the policy administrator and uses the knowledge it acquires from detected abnormal accesses to generate access policy adjustments at runtime. APA replaces the PAP component of XACML and consists of the the following sub-components: the access behavior classifier and the access policy adjuster. In the following, we describe these component.

1. **Access Behavior Classifier**

   An access behavior represents the manner in which a user utilizes the system resources. We define the access behavior as the set of access requests submitted by the user to the authorization server within a predefined time window. Formally, we define the access behavior, denoted by AB, as follows:

$$AB = \{AR_{t-k}, AR_{t-k+1}, \ldots, AR_t\}$$

where AR_{t-k} is the first access request in the access behavior AB in a time window k. Each access request is represented by the set of attributes that characterize the elements of the access request (i.e., user, resource and operation) and the context information that describes the environment in which the access request takes place (e.g., time and location). Formally, we define the access request, denoted by AR, as follows:

$$AR = \{AT_u, AT_r, AT_o, AT_c\}$$

where $AT_u = \{at_1^u, ..., at_x^u\}$ is the set of user attributes, $AT_r = \{at_1^r, ..., at_y^r\}$ is the resource attributes, $AT_o = \{at_1^o, ..., at_z^o\}$ is the operation attributes, and $AT_c = \{at_1^c, ..., at_w^c\}$ is the context or environment attributes. An attribute is expressed as $at = name\ op\ value$, where op is a relational operator (e.g.,$=, \neq, <$) between the attribute name and a value from the range of possible values of this attribute.

We define an abnormal behavior, denoted by AB_0, as the AB that consists of one or more access requests that violates administrator-defined proactive measures (e.g., frequent access denials, resource overuse).

The access behavior classifier performs two tasks: (1) the offline training, during which the classifier models the access behaviors based on the historical access information submitted by the CPE component (i.e., user access logs) and administrator-defined proactive measures; (2) the access behavior classification, during which the classifier component classifies incoming access requests and reports abnormal access behavior(s) to the access policy adjuster component.

Listing 3.3: Access Policy Adjustment Algorithm.

```
input  : Abnormal Access Behavior $AB_0$, Fact Base $FB$,
         Proactive Measures $D$, adjustment threshold $TH_{adj}$
output: adjustments $ADJ$
begin
    $ADJ \leftarrow \{\}$   // initialize
    for $Req \in AB_0$
        for $A \in Req$
            Calculate $Pr(D|A)$
            if $Pr(D|A) \geq TH_{adj}$
                $Query.target \leftarrow \{A.key, penalty\}$
                  $ADJ \leftarrow ADJ \cup \{Query\}$
            end
        end
    end
    $ADJ \leftarrow \{\textbf{Consult\_Admin}(ADJ)\}$
    for $Fact \in FB$
        for $Query \in ADJ$
            if $Query.key \subseteq Fact.target$
                $Fact.update(Query.value)$
            end
        end
    end
    Notify(CPE)
end
```

2. **Access Policy Adjuster**

Once the classifier model is built, we need to apply the knowledge it acquires during the training phase to adjust the access control policies. Listing 3.3 shows the access policy adjustment algorithm.

The adjustment algorithm takes the abnormal access behavior reported by the classifier and the administrator-defined proactive measures as inputs, and returns a set of access policy adjustments in the form of primitive facts update queries. The algorithm performs four steps:

(a) It extracts the set of attributes that describe the subject, resource, operation, and environment associated to each request in the abnormal access behavior, and for all combinations of these attributes, it calculates the probability of a resource misuse given that the attribute (or combination of attributes) A is present in the abnormal behavior AB_0, lines 3-5.

(b) It evaluates the calculated probabilities against an administrator-defined adjustment threshold, and generates new policy, lines 6-8, in the form of update queries. The update query is defined on the attribute(s) that contribute most to the abnormal access behavior. The query target consists of a key-value pair. The query key is set to the value of the attribute key or combination of keys; specifying to which access element(s) of a primitive fact, the update query applies (e.g., subject, operation, object, environment or a combination of these). The query value determines the penalty of resource misuse.

(c) It consults the policy administrator and gets the final approved policy adjustments, and updates the FB by updating all primitive facts that contain a key-value pair that matches the query target. Lines 12-18.

(d) It notifies the CPE of the policy update line 19.

We calculate the conditional probability, line 5, as follows:

$$Pr(D|A) = \frac{Pr(A \cap D)}{Pr(A)} \tag{3.1}$$

Where:

$Pr(D|A)$ represents the contribution of the attribute A to the the abnormal behavior AB_0.

$Pr(D)$ is the probability of resource misuse in AB_0, defined as follows:

$$Pr(D) = \frac{\#\ of\ misuse\ incidents}{k} \tag{3.2}$$

$Pr(A)$ is the probability of an attribute A to present in AB_0, defined as follows:

$$Pr(A) = \frac{\#\ of\ A\ presences\ in\ AB_0}{k} \tag{3.3}$$

$Pr(A \cap D)$ is the probability of an attribute A to present in a resource misuse incident in AB_0, defined as follows:

$$Pr(A \cap D) = \frac{\#of\ A\ presences\ in\,misuse\ incidents}{k} \tag{3.4}$$

k is the number of access requests in AB_0.

The adjustment threshold TH_{adj} is a predefined application-dependant value that is set by the policy administrator. The evaluation of the calculated probability can result in the generation of one of two types of policy adjustment:(1) revocation adjustment, which applies only to the element association type of primitive facts ; or (2) limitation adjustment, which applies to the guard context type of primitive facts. Both types of policy adjustments can be defined to adjust primitive facts based on single or combined access elements. The former is a recommendation for the policy administrator to adjust primitive facts based one access element (e.g., subject), while the latter is a recommendation for policy administrator to adjust primitive facts based on two or more access elements (e.g., subject and object).

Setting TH_{adj} to a small value increases the probability of the adjuster component to pose more restrictions on the access policies. For example, if we set TH_{adj} to a value that is less than $Pr(A \cap D)$, the adjuster component will always recommend to adjust all primitive facts that contain the attribute A.

For attributes that have categorical values such as the names of core access elements and locations, we set the value of misuse penalty to `Null`. For numerical attributes values such as the time, we define the penalty, denoted as P_0, as a function of the attribute contribution to the abnormal behavior as follows:

$$P_0 = A.value - (Pr(D|A) - TH_{adj})(A.value) \tag{3.5}$$

For example, if the probability of the attribute (`subject-id: "Adam"`) to present in a misuse incident, in an abnormal behavior, is greater than or equals to (TH_{adj}), the adjuster component generates an update query and sets the query target as follows:

$$Query.target \leftarrow \{subject - id : \text{“}Adam\text{”}, Null\}$$

If the update query is approved by the policy administrator, the attribute (`subject-id:"Adam"`) is updated with (`subject-id:"Null"`) in all element associations whose subject is Adam, thus all access permissions that subject Adam has on the system resources will be revoked.

Assume that the adjustment threshold (TH_{adj}) equals 0.5 and the probability of the combined attributes (`subject-id:"Adam", time:"10:00-11:00"`) to present in a misuse incident is 0.75, the adjuster component generates an update query and sets the query target as follows:

$$Query.target \leftarrow \{subject-id : \text{“}Adam\text{”}, time : \text{“}10:00-11:00\text{”}, \\ time : \text{“}10:00-10:45\text{”}\}$$

If the update query is approved by the policy administrator, the combined attributes (`subject-id:"Adam", time:"10:00-11:00"`) is updated with (`subject-id:"Adam", time:"10:00-10:45"`), thus the time frame during which Adam can access the system resources is reduced by 15 minutes.

3.4 DACIoT Architecture

Figure 3.3 depicts our proposed access control architecture for DACIoT. The architecture consists of three interacting entities: the Authorization Server (AS), the accessing user and the target IoT device. AS is the entity that decides which IoT resources a user can access, by integrating the functionalities of the APS, CEP, and PAP. The implementation and placement of the AS may vary depending on the scale, latency constraints, and number of users/devices in a specific area. An area of intense users/IoT devices deployment may require multiple servers, each serving a cluster of resources/users. Deciding whether to locate the APS centrally to support multiple CEPs, to dedicate one APS to each CEP, or to adopt a hybrid of the two approaches can be associated with various latency and performance features. In particular, the placement of CEP (e.g., at the application side, at the resource manager side, or at

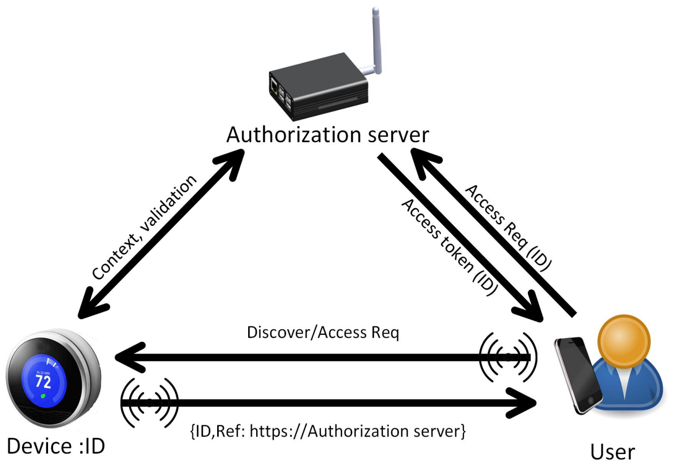


Figure 3.3: DACIoT Architecture.

the proxy) can contribute to how efficiently the authorization server handles a large number of access requests.

For these reasons, we chose to design a unique decentralized proxy-based access control architecture and we placed all AS components on the network edge. The proxy devices can be WI-FI access points, smartphones or other edge node devices with sufficient computational resources and battery capacity. Our designed architecture brings many advantages including:

- Technical advantages: The proxy device provides the computation and communication capabilities that IoT devices cannot support for resource demanding computations, such as the automatic access policy specification and context monitoring. In addition, it lowers end-to-end latency required for continuous access policy enforcement, by having the access decision made close to where access requests are originating and real-time context information is provided. Further, placing the APS and CEP on the same machine within the trust domain of device owner or administrator, removes the need for securing the communication channel between the two components.

- Non-technical advantages: decentralization preserves data privacy; it provides device owners or administrators with the ability to exert direct control over their data, rather than allowing it to be manipulated or shared by third parties. The architecture allows the data owner to determine who can access which device and for what reason before the data leaves their trusted domain. With decentralized access control, device owners can define privacy-aware access control policies that allow the data generated by their devices to be released to the cloud only in the form that comply with their privacy requirements. For example, businesses that provide daycare services to children and infants, can define access rules that blur faces in images and videos. Other examples of privacy-aware access policies include data summarization or aggregation. For instance, access rules can be defined to release daily or weekly totals of a building energy usage rather than per-minute measurements. Other situations may involve highly sensitive information. For instance, in a smart home regular readings from home sensors could provide significant insights into the behavior of the occupants such as their presence in the house and daily activities. In such cases, access rules can be defined such that sensor readings are coarsely aggregated or omitted at certain times.

Figure 3.4 shows the time sequence diagram that illustrates how the different components of DACIoT interact with each other to control access to IoT resources and maintain up-to-date access control policies during the lifetime of an access session.

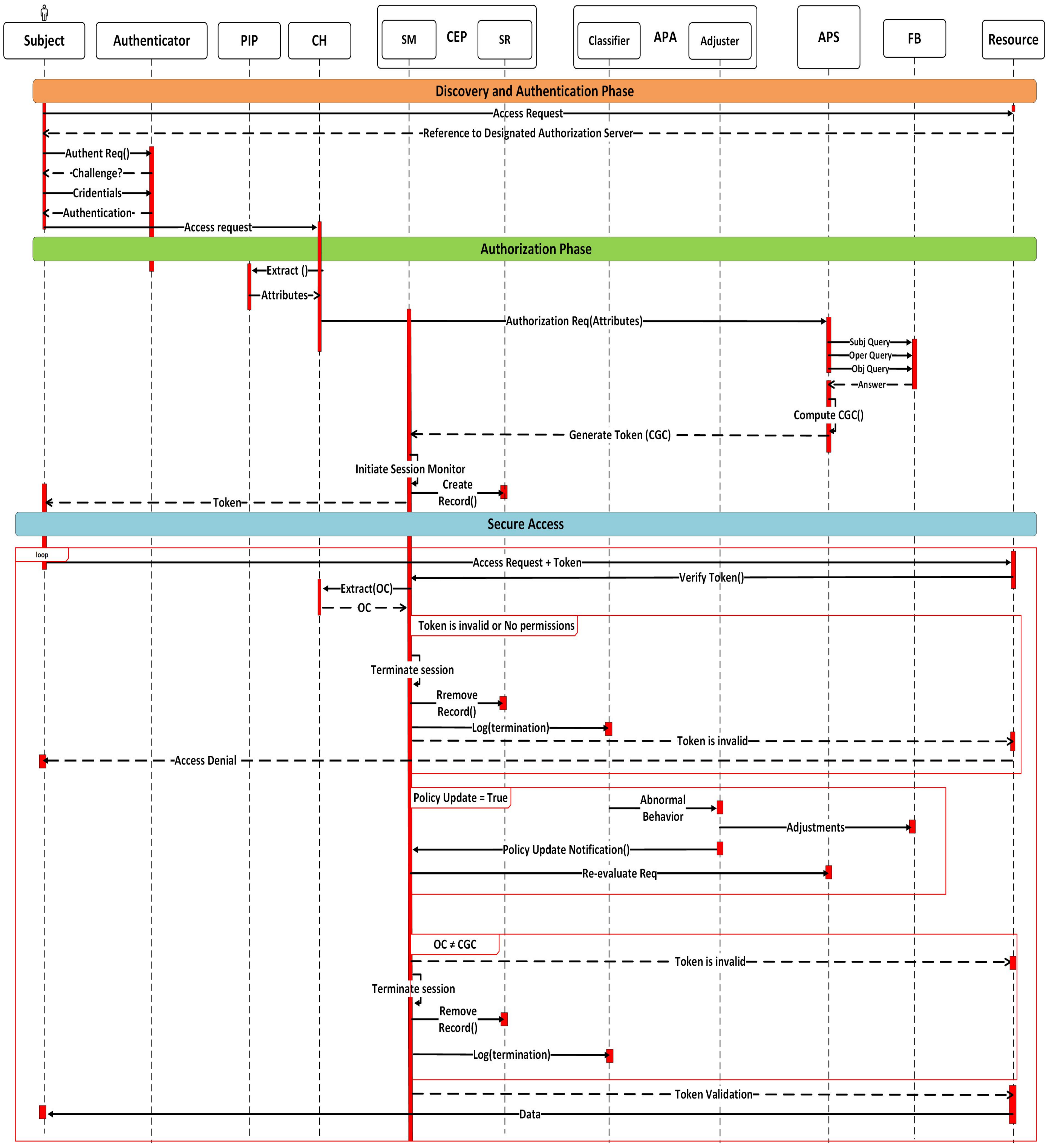


Figure 3.4: DACIoT Work Flow.

The access control goes through three operational phases: discovery and authentication, authorization, and secure access. The mechanisms of device discovery and user authentication are beyond the scope of this research. Therefore, we assume that their functionalities are in place. The access starts by the requesting entity (i.e., Adam's smartphone) discovering resources in its proximity. We assume that Adam has a smartphone that supports Bluetooth low energy technology (BLE) [107] to scan and discover available IoT devices and services. If a resource of interest is publicly discoverable, the resource responds with its profile and the reference to the designated AS. Adam then submits an access request to AS. AS replies with a challenge asking the accessing entity for access credentials. If the requesting entity is authenticated, the AS evaluates the access request based on the attributes of the requesting entity, requested resource, and environment and sends the decision along with the set of access permissions back to the requesting application in form of self-contained access token. The user application attaches this token to every request to the protected IoT device. The IoT device verifies the access token every time a new access request is submitted. If the token is valid and the user has permissions, the IoT device allows access to its resources based on the set of permissions and access expiration time embedded in the access token. Otherwise access is denied.

3.5 Summary

Access control is a key enabler for the adoption of IoT technology. The proliferation of IoT devices in our surroundings collecting and disseminating sensitive information raises considerable security and privacy concerns. The highly dynamic nature of IoT environment poses unique access control challenges rendering standard access control policies, models, and mechanisms unsuitable for to IoT scenarios. In this chapter, we identified the limitations of current access control approaches and the requirements of a robust and reliable access control mechanism that can efficiently control access to IoT devices and dynamically adapt to frequent changes in access contexts.

This chapter presents DACIoT a dynamic access control framework for IoT environments. DACIoT improves access policy management by enabling the generation and adjustment of conflict-free and context-aware access control rules at run-time. DACIoT ensures continuous authorized access to IoT devices at the time of access request; as well as during the lifetime of an access session taking into consideration dynamic changes in the access control policy and the environmental context in which the access takes place. DACIoT components are presented using the XACML access control reference model. A comprehensive description is provided for each component along with the possible approaches from existing technologies that could potentially implement such a component. Our proposed decentralized access control architecture enables DACIoT to evaluate access requests, monitor access sessions and adjust access control policies based on real-time context information, which helps involve IoT devices in access decision-making while not introducing considerable delays to access decision enforcement.

Chapter 4

DACIoT Validation and Performance Evaluation

4.1 Introduction

In this chapter, we validate the DACIoT framework and evaluate it's performance. We provide a use-case scenario to discuss the functional requirements of DACIoT and explain how the proposed framework can be applied. We design and implement a prototype of DACIoT to apply our use-case scenario in a real-world IoT environment. To validate DACIoT, we use our prototype to reproduce XACML access control policies, and use the following properties to evaluate the resultant DACIoT policies:

- Correctness: checks if DACIoT access control policies leak access permission to unauthorized or unintended users.
- Consistency: ensures that DACIoT access control policies are conflict-free.
- Completeness: assures that each access request is either accepted or denied by DACIoT access control policies.
- Context-awareness: assures that DACIoT access policies are sensitive to changes in access contexts and can make different access decisions in different access

contexts.

For performance evaluation, we carry out various experiments and use the following metrics to assess the performance of DACIoT:

- Access Response Time: is the time that the DACIoT requires to evaluate an access request and enforce the access decision.
- Re-evaluation Time: is the time DACIoT takes to re-evaluate all active access sessions in response to changes in the access policies.
- Access Behavior Classification Accuracy: is a measure of how efficiently DACIoT can differentiate between normal and abnormal access behaviors.
- Policy Adjustment Accuracy: measures the improvement in the protection of system resources that DACIoT policies achieve over the original policies defined by a system administrator.

The remainder of this chapter is organized as follows. Section 4.2 provides a use-case access scenario involving a student accessing university resources; it describes how DACIoT should respond under various access contexts. Different access scenarios are applied to test the correctness, consistency, and completeness of DACIoT access control policies. Section 4.3 describes our developed prototype of DACIoT as well as the experimentation environment. Section 4.4 presents the performance evaluation of the proposed framework; including a detailed description of the conducted experiments and an in-depth discussion of the obtained results. Section 4.5 summarizes the experimental results and concludes the chapter.

4.2 Use-case Scenario

Suppose Adam, a graduate student at the School of Computing (SoC), at Queen's University, is meeting his supervisor Eve in the School's conference room. According to Eve's calendar, the meeting is scheduled for one hour (e.g., 10:00 - 11:00) every week. When Adam arrives on campus, based on his current context (i.e., student's profile, location, and time), DACIoT grants Adam access to a few resources campus-wide such as smart-parking, Wi-Fi network, and the main entrance to the school's building. When Adam enters the building, DACIoT considers the change in Adam's context and dynamically assigns additional access privileges to Adam, based on the new fact (i.e., his new location), including entrance to the conference room. While waiting for Eve in the conference room, Adam may not have privileges to access the conference room facilities. However, when Eve arrives, Adam's context changes again so that during the meeting time, DACIoT considers the coexistence of both Eve and Adam in the conference room and allows Adam to access conference room-specific resources (e.g., e.g., indoor environmental controls (HVAC)). Suppose that Eve has another meeting that starts at 11:00. When Eve leaves the conference room, DACIoT considers the change in Adam's context and immediately revokes all access permissions Adam has to the conference room-specific resources. This simple scenario illustrates how DACIoT dynamically and continuously controls access to university resources.

The use-case fits the requirements of the validation and performance evaluation of our proposed framework for the following reasons:

1. The use-case represents a real-world IoT environment where resource-limited devices (i.e., smart keys, door locks, microcomputers) interact to control access

to the internal and external doors of the school building.

2. The use-case employs context-aware access control policies that control access to school resources based on time and location. This allows us to test the context-awareness and dynamic authorization aspects of the proposed framework.

3. The use-case involves access control policies that define multi-level authorization and hierarchies. For example, in order to control the HVAC system in the conference room (authorization level 3), first, Adam needs to have access permissions to access the school building (authorization level 1), Second, he needs to access the conference room (authorization Level 2). This allows us to test the dynamic authorization aspect of the proposed framework.

4. The use-case involves access scenario where access context changes while a resource is in use. This allows us to test the continuous policy enforcement aspect of our proposed framework.

5. The use-case provides us with a real-life data (i.e., access logs) collected over four years during which the access policies have been updated many times to meet changes in security requirements of the school such as adding/revoking access keys, assigning/ revoking specific keys on specific doors and so on. In addition, we were able to investigate the access policies that generated the access logs considered in the use-case and directly communicate our results with the policy administrator. All of which allows us to test the adaptive policy adjustment aspect of our proposed framework.

4.2.1 DACIoT Policy Specification

Listing 3.4 shows the access control policy defined in XACML language to control access to the university resources considered in this access scenario.

Listing 4.4: XACML Access Control Policy to SoC Resources

```
  PolicyCombiningAlgId= 'Permit-Overrides'>
<Target/>
<Policy PolicyId= 'SoC resources'
  RuleCombinationAlgId= 'First-Applicable'>
<Target/>
<Rule RuleId='1' Effect='Permit'>
<Target>
<Subjects>
<Subject> faculty-member </Subject>
<Subject> staff </Subject>
<Subject> grad-Student </Subject>
</Subjects>
<Resources>
<Resource> smart-park </Resource>
<Resources><Resource> main-entrance </Resource>
<Resources><Resource> wi-fi </Resource>
</Resources>
<Actions>
<Action> implied-action </Action>
</Actions>
</Target>
</Rule>
<Rule RuleId= '2' Effect= 'Permit'>
<Target>
<Subjects>
<Subject> supervisor </Subject></Subjects>
<Subjects>
<Subject> grad-student </Subject></Subjects>
<Resources><Resource> printer </Resource>
</Resources>
<Actions><Action> implied-action </Action>
<Environments>
<Environment>Location:conf-room <Environment>
<Environment> Time:10:00-11:00 <Environment>
</Environments>
</Target>
</Rule>
</Policy>
```

```
39</PolicySet>
```

The policy includes two access rules: Rule 1, lines 6-21, states that a faculty member, staff or a graduate student can reserve a parking spot in the school smart-parking lot, unlock the main entrance and connect to the school's Wi-Fi network; Rule 2, lines 22-36, states that a graduate student can control the temperature (HVAC) in the conference room only in the presence of their supervisor and within the time frame 10:00-11:00.

Table 4.1: Primitive Facts for Access Rule 1.

PF_1	`{ type:"subject", subject-id:"Adam", role:"grad-stu", object-id:"smart-park"}`
PF_2	`{ type:"operation", operation-id:"reserve", subject-id :"Adam", role:"grad-stu"}`
PF_3	`{ type:"object", object-id:"smart-park", operation-id:"reserve" }`
PF_4	`{ type:"subject", subject-id :"Adam", role:"grad-stu", object-id:"Wi-Fi"}`
PF_5	`{ type:"operation", operation-id:"connect", subject-id :"Adam", role:"grad-stu"}`
PF_6	`{ type:"object", object-id:"Wi-Fi", operation-id:"connect" }`
PF_7	`{ type:"subject", subject-id :"Adam", role:"grad-stu", object-id:"main-entrance"}`
PF_8	`{ type:"operation", operation-id:"unlock", subject-id :"Adam", role: "grad-stu"}`
PF_9	`{ type:"object", object-id:"main-entrance", operation-id:"unlock" }`

Table 4.1 shows the primitive facts that DACIoT generates to give the same effect of access Rule 1 defined for graduate students in this use-case scenario. Note that no environmental (or contextual) conditions are defined in access Rule 1. Therefore, the corresponding primitive facts simply define the subject-object, operation-subject and object-operation associations for each object. For example, DACIoT generates PF_1 to PF_3 to control the access of the graduate students to the smart-parking object.

Access Rule 2 is an example of a context-aware access rule; it defines three contextual conditions a subject needs to satisfy to gain access to the protected object. The contextual conditions define restrictions on the subject location, request time and subjects coexistence referring to the collocation of subjects. We assume that the policy administrator uses the XACML policy editor to define the university access control rules during system setup time. Table 4.2 shows the primitive facts that DACIoT generates to give the same effect of the access Rule 2 defined for graduate students.

Table 4.2: Primitive Facts for Access Rule 2.

PF_{10}	`{ type: "subject", subject-id:"Adam", role:"grad-student", object-id:"HVAC", time:"10:00-11:00", location:"conf-room", coexistence:"true" }`
PF_{11}	`{ type: "operation", operation-id: "control", subject-id :"Adam", role: "grad-student", time:"10:00-11:00", location:"conf-room", coexistence:"True"}`
PF_{12}	`{ type: "object", object-id: "HVAC", operation-id:"control", time:"10:00-11:00", location: "conf-room" }`

In PF_{11}, the key `coexistence` refers to the collocation of both Eve and Adam in the conference room.

4.2.2 DACIoT Policy Enforcement

An innovation in DACIoT is that, after an access request is permitted, the AS continuously checks the operational context of the requesting and the requested entities. If the operational context does not satisfy the guard context specified in the access token, the access token is deemed invalid, and the access session is terminated immediately. For example, if Eve leaves the conference room at 10:30, all access tokens that Adam has obtained during the meeting time continue to be valid time-wise until 11:00, however, if Adam submits an access request, for instance, to control the HVAC, the HVAC verifies the access token associated to Adam's request through the SM module. The SM module in turn extracts the current OC of Adam as well as the HVAC through the CH, and finds that the OC of Adam has changed (i.e., coexistence is False) and does not comply to the guard context specified in the access token. As a result, the SM terminates the access session, returns an access token invalidation to HVAC. The HVAC in turn denies all subsequent access requests that use this access token.

4.2.3 DACIoT Policy Adjustment

To elaborate on how DACIoT handles automatic policy specification and adjustment, suppose that the access logs generated by the SM module show that Adam, either accidentally or intentionally, attempted to access other school resources (i.e., an office or lab room) he is not authorized to have access to without plausible reasons. Although the university access policy would deny Adam's access requests, this scenario may indicate that Adam is either abusing his access privileges for personal benefits or his access key is being used by an attacker.

When Adam's access denials exceed, for instance, the threshold defined by the policy administrator for graduate students, the classifier module classifies Adam's access behavior as abnormal and triggers the adjuster module to provide the appropriate countermeasures. The adjuster module analyzes the abnormal access behavior of Adam and finds, for instance, that Adam's identity attribute (i.e., subject-id:"Adam") contributes most to the abnormal behavior with a probability of 95%, followed by Adam's role attribute (e.g., role: "grad student") with probability of y%, Adam's environmental attributes (e.g., location) with probability of z% and so on. Based on the administrator-defined adjustment threshold (e.g., 0.75), the adjuster module recommends revoking all access permissions assigned to Adam until his subject identity is reviewed and access privileges reinstated. In such a case, all policy adjustments are defined based on the subject identity (i.e., Adam). If the policy administrator approved the recommended policy adjustments, the adjuster module sets the subject-id to `Null` in all primitive facts in which the subject-id value is "Adam" (i.e., PF_1, PF_2, PF_4, PF_5, PF_7, PF_8,PF_{10}, PF_{11}), and notifies the SM module of the primitive fact update.

The SM does not interrupt Adam's other ongoing access sessions (e.g., Wi-Fi connection), rather the SM re-evaluates these access sessions against the up-to-date primitive facts. For each ongoing access session that Adam has, the SM submits a re-evaluation access request to the APS module. The APS considers the re-evaluation access requests as new access requests and queries the FB searching for primitive facts that match the elements of these access requests. The APS updates the CGC for each re-evaluated access session, which is `null` in this case (i.e., Adam has no permissions), and sends it back to the SM module. The SM module immediately terminates the corresponding access session (i.e., all access sessions that belong to Adam in this case).

Another interesting scenario is when the adjuster module finds that most of Adam's access denials took place in specific contexts. For example, 85% of Adam's access denials were in the conference room while waiting for his supervisor. In such cases, the adjuster module recommends limiting Adam's access permissions by updating all primitive facts in which the subject is Adam, and the location is the conference room (i.e., PF_{10} and PF_{11}). This scenario results in partial revocation of access permissions assigned to Adam such that Adam is denied access to the resources in the conference room only, but can still access other university resources considered in this use-case scenario.

4.3 Experimentation Environment

To validate the functionality of DACIoT, we developed a dynamic access control prototype in an IoT environment. We carried out several experiments to validate DACIoT operation to ensure that it functions as expected. The prototype consists of two parts: the client and the authorization server. The client part is a mobile-based interface that supports two operational modes: user and administrator. The user interface enables the IoT users to search for and connect to IoT devices within their proximity, in addition to the designated authorization servers of the device. The administrator interface enables the policy administrator (or device owner) to define and manage access control policies for their IoT devices. This part of the prototype is deployed on a Samsung Galaxy S7 smartphone (Quad-core 2.2 GHz Snapdragon 820, 32 GB RAM, Super AMOLED 2560x1440 pixels display, 5.1 inches) with a rooted Android 6.0 Marshmallow platform [108], connected to a Wi-Fi network. As well, we assume that users in this case scenario use a smart key to access the external and internal doors of a university building.

The authorization server part is deployed on a Raspberry Pi 3 (quad-core 1.2 GHz ARM Cortex A53, 1 GB SDRAM, 10/100 MBPS Ethernet, 802.11n Wireless LAN, Bluetooth 4.0) [109] and implements the authorization functionalities introduced in this paper: APS, CPE, and APA. In addition, it handles device registration (i.e., authentication and attribute management) as well as context monitoring. For policy management, we structured the primitive facts in a K-dimensional tree (k-d tree) [76] to achieve faster primitive facts matching, incur less overhead to access policy enforcement and facilitate runtime policy adjustment. We chose the k-d tree data structure because of its usefulness in applications that involve a multidimensional search key over large scale datasets. We built three separate k-d trees for the subjects, objects, and the operations offline. For authentication, we authenticate users based on their cell phone numbers. For attributes management (i.e., PIP), we use a simple form of a database to store the subject, object, and operation attributes in a CSV formatted excel file and retrieve these upon access request evaluation. For environmental attributes, we consider two: subject location and request time. The subject location is monitored based on Bluetooth proximity to the authorization server, and time of request is determined based on server local time. For IoT resources, we use two commercial IoT devices in this prototype: a Sensor Tag [110] device representing an IoT device that continuously generates data such as room temperature and a WeMo smart switch [111] representing an IoT device that can be actuated. We use the smart switch and sensor tag devices to replace the mechanical door lock and Wi-Fi connectivity resources, respectively.

4.3.1 Correctness of DACIoT Policies

To ensure the correctness of the access control policies that DACIoT generates, we used the DACIoT administrator interface to create the access control policy described in Listing 4. Next, we used the DACIoT user interface to submit access requests to the DACIoT server and observed the results. To ensure test coverage, we generated a set of test requests that encompasses all permutations of attribute values considered in our use-case. To simplify the generation of test requests we assumed that the operational context of an access request either satisfies or does not satisfy the contextual conditions (e.g., Adam is either in or out of the conference room). The total number of test requests is calculated as follows:

2 `subject-identities` X 2 `subject-roles` X 4 `objects-identities`
X 4 `operations-identities` X 2 `location values` X 2 `time values`
X 2 `coexistence values` = 512 test requests.

Finally, we checked DACIoT access decisions manually and found that DACIoT correctly evaluated all test requests.

4.3.2 Consistency of DACIoT Policies

A policy or rule conflict occurs when access policies (or rules) grant in-consistent access permissions to an accessing entity [112]. Detecting policy (or rule) conflicts is challenging; it involves detecting of the conflicting policies (or rules) and identifying the type of conflict among policies (or rules) at runtime. Also, it requires dynamic verification to ensure conformance with security requirements specified by access policies. For example, XACML defines four policy or (rule) combination algorithms including, *First-Applicable*, *Only-One-Applicable*, *Deny-Overrides*, and *Permit-Overrides*. These algorithms resolve conflicts when an access request applies to more than one policy or (rule). For a First-Applicable policy or (rule), the decision of the first applicable policy or (rule) is returned. For an Only-One-Applicable policy or (rule), the decision of the only applicable policy or (rule) is returned; `Indeterminate` (i.e., conflict or error) is returned if there are more than one applicable policy or (rule), or if one or more attribute(s) is missing. For a Deny-Overrides policy or (rule), `Deny` is returned if any policy or (rule) evaluation returns deny; `Permit` is returned if all policy or (rule) evaluations return permit. For a Permit-Overrides policy or (rule), `Permit` is returned if the evaluation of any policy or (rule) returns permit; `Deny` is returned if the evaluations of all policy or (rule) return deny. If an access request applies to none of these algorithms `Not-Applicable` is returned.

However, DACIoT requires no policy combination algorithm; DACIoT follows the white list authorization approach where all access requests are denied except those defined in the white list in the form of primitive facts. DACIoT removes the burden of detection and resolution of access policies (or rules) conflicts and always generates consistent access policies because DACIoT does not store access policies or rules permanently, rather it generates them on the fly based on the always up-to-date primitive facts. Therefore, a change in one fact would automatically affect all possible access rules that can be generated based on that one fact.

4.3.3 Completeness of DACIoT Policies

Policy completeness assures that each access request will be either accepted or denied by the access control policy. For example, if an attribute is missing from an access request, XACML policy does not make a final access decision, rather it shifts into an indeterminate state, which requires further processing (i.e., a policy or rule combining algorithm). Unlike XACML policies, if DACIoT receives a request with a missing attribute, a subject attribute for instance, the APS will search the FB for a primitive facts that exactly match the three access elements of this request and can get into one of two states that both leads to an access denial: (1) No primitive fact is defined for the subject with missing attributes, as such, access is denied; (2) One or more primitive facts are defined for the subject element with a missing attribute. Yet, due to missing attribute, the subject can only match the requested operation or the requested object but not both; hence, access is also denied.

4.3.4 Context-awareness of DACIoT Policies

In this experiment, we test the ability of DACIoT access control policies to respond correctly to changes in the time and location of access requests. To test the DACIoT response to changes in the time of access request, we use our prototype to generate two access requests to actuate the smart switch emulating a student access request to the door lock of the conference room. The access requests are submitted to DACIoT authorization server at 10:20 and 11:20 respectively. The first request was permitted by DACIoT, while the second request was rejected. These results show that DACIoT can consider the time of access requests in access decision-making and return correct responses based on the predefined access control policies. To test the DACIoT response to changes in the location of access requests, we define an access policy that assigns two levels of authorization on the sensor tag device emulating the Wi-Fi connectivity in the school of computing. According to this policy, devices that are in the bluetooth range of the authorization server can read the temperature value every two minutes, while devices that are out the bluetooth range can read the temperature value every ten minutes. We use the client application on two smartphones, located in and out of the bluetooth range of the authorization server, to generate two access requests to read the room temperature from the sensor tag. For the access request that is submitted by the smartphone within the bluetooth range of the authorization server, our application was able to read the temperature value every two minutes. For the access request that is submitted out of the bluetooth range, our application was able to read the temperature value every ten minutes. These results show that DACIoT is sensitive to user location and can assign different access permissions when the user location changes.

4.4 Performance Evaluation

To evaluate the performance of the DACIoT, we conducted several experiments that investigate various aspects of DACIoT, including, access response time, average re-evaluation time, access behavior classification accuracy, and policy adjustment accuracy. Experiments show how DACIoT can adapt to application security and time-sensitivity requirements offering three re-evaluation strategies. Experiments also demonstrate how DACIoT can maintain up-to-date access control policy recommending precise policy adjustments based on highly accurate access behavior classification.

4.4.1 Access Response Time

In our experiments, we define the access response time as the time that DACIoT requires to evaluate an access request and enforce the access decision, including, access request analysis, attribute extraction, and primitive fact matching. Figure 4.1 depicts the access response time of DACIoT and XACML access control engines for 100 concurrent access sessions versus the varying number of access rules. We compare the performance of DACIoT and XACML because XACML controls access based on the ABAC model that generalizes and extends earlier access control models. ABAC is considered the most flexible model in terms of policy specification and enforcement. In our experiments, we show that DACIoT generates more flexible policies than XACML and provides better security and adaptability to highly dynamic IoT environments.

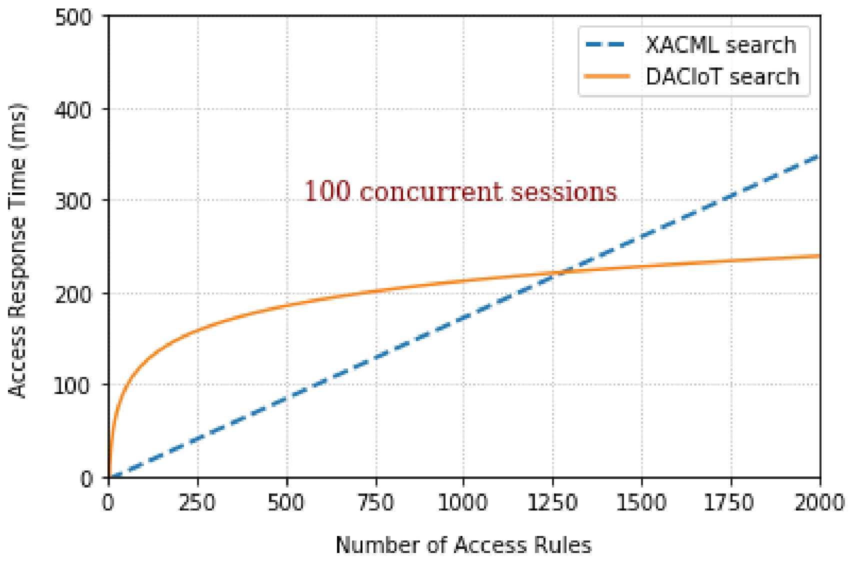


Figure 4.1: DACIoT Access Response Time.

To evaluate an access request the XACML access control engine checks the attributes of the access request against the access policy or policy set. If a request satisfies the target of a policy, then the request is further checked against the rule set of the policy; otherwise, the policy is skipped without further examining its rules. The same applies to the target of the policy set. The high complexity of XACML policies in terms of policy hierarchy and rule conflicts makes linear searching (i.e., brute force) appear to be the natural way of processing access requests.

The time complexity of DACIoT is simply bounded to the time required to search each of the subject, object, and operation k-d trees. Assuming the three k-d trees are of the same size n, and the fact that k-d tree is a special case of binary trees, DACIoT has logarithmic time complexity T which is given as follows:

$$T(n) = 3 * N * O(\log(n)) \tag{4.1}$$

Where N is the number of access rules. When $n >> 2^k >> N$, where k is the number of attributes and context constraints that describe each type of the core access control elements, the time complexity can be further simplified to:

$$T(n) = O(\log(n)) \tag{4.2}$$

For each number of access rules, we calculate the access response time of DACIoT and XACML over 10 runs and take the average. Results show that at a low number of access rules, the access response time of DACIoT tends to increase rapidly when the number of access rules increases. This increase in time is because the k-d tree search needs to visit relatively more tree branches for a lower number of points (i.e., access rules). However, when the number of access rules increases significantly, DACIoT search outperforms XACML search. For example, DACIoT is 60 ms faster than XACML at 1750 access rules. These results show that our approach can scale efficiently to large scale environments such as IoT.

4.4.2 Re-evaluation Time

Re-evaluation time is the time DACIoT takes to re-evaluate all active access sessions in response to changes in the primitive facts, more specifically, it is the time elapses between the detection of changes to the primitive facts and the completion of re-evaluation of all ongoing access sessions against the changes to the facts. DACIoT supports three re-evaluation strategies that cater to various security and time sensitivity requirements of different business applications. The re-evaluation strategies include: *Re-evaluate and Decide*, *Stop and Re-evaluate*, and *Hybrid*.

1. **Re-evaluate and Decide** supports application domains in which resource utilization is a priority than limited unauthorized access. Following this strategy, DACIoT does not interrupt ongoing access sessions; rather it first re-evaluates them against the access control policy and second, based on re-evaluation results, DACIoT only stops the denied access sessions. The strategy maximizes resource utilization giving a chance for users whose access sessions might not be affected by the policy changes to continue seamless access to protected resources. In addition, it incurs less re-evaluation time than other strategies because DACIoT only needs to terminate the effected access sessions, and no further processing is required. However, this strategy compromises system security because it may expose system resources to unauthorized access during the re-evaluation time.

2. **Stop and Re-evaluate** strategy supports application domains in which system security is a priority. Following this strategy, DACIoT stops all ongoing access sessions when any changes in access context are detected, reserves their parameters, and re-evaluates the access sessions against the access control policy. Based on the re-evaluation results, DACIoT restores permitted access sessions. This strategy provides more secured access to system resources than other strategies at the expense of limited interruption. With this strategy, however, both system and users are subject to opportunity costs because legitimate users are blocked from utilizing system resources during re-evaluation time. Also, the strategy

incurs more re-evaluation time re-establishing non-affected access sessions.

3. **Hybrid** strategy takes advantage of previous strategies and overcomes their limitations. Following this strategy, DACIoT switches between re-evaluation strategies based on application-dependent switching threshold. Several dynamic factors can be considered to calculate the switching threshold, including the user access history, resource sensitivity, and system threat-level, to name a few. This strategy makes a fair compromise between resource utilization and security; it enables users to utilize system resources in normal access contexts fully and maintains sufficient system security in critical or unexpected access contexts.

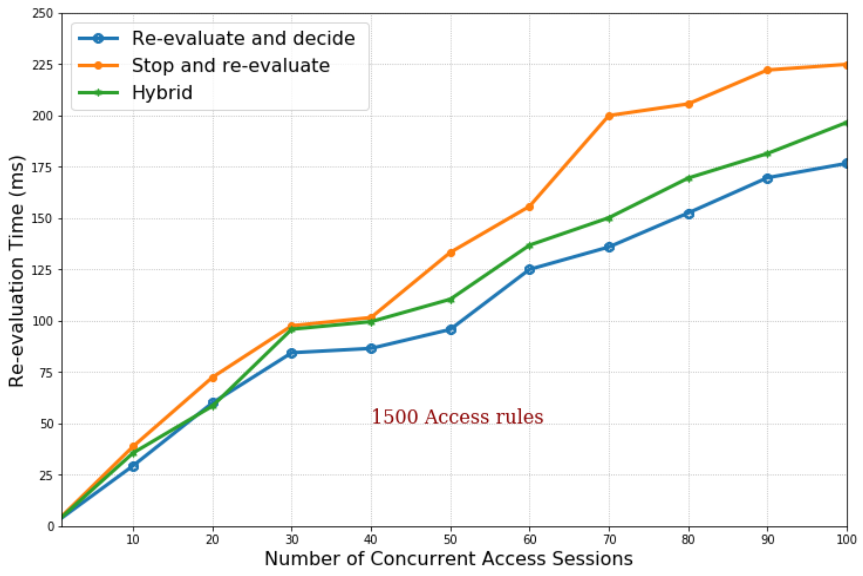


Figure 4.2: DACIoT Re-evaluation Time.

Figure 4.2 depicts the actual re-evaluation time of DACIoT versus a variable number of concurrent access sessions. The figure shows that the re-evaluation time for the three strategies is proportional to the number of concurrent access sessions and tends to increase slowly when the number of concurrent access sessions increases.

In this experiment, we use a random variable with normal distribution to generate the switching threshold values for the hybrid strategy.

We apply linear regression to obtain the estimated time for DACIoT re-evaluation. Table 4.3 shows the slope and intercept values for the estimated re-evaluation time of the three strategies. The slope values show that for every additional access session, the re-evaluation time increases by an average of 1.69, 2.28 and 1.86 milliseconds for the Re-evaluate and Decide, Stop and Re-evaluate and Hybrid strategies respectively. The estimated re-evaluation time is calculated using the following equation:

$$Time_{EST} = b_0 X + b_1 \tag{4.3}$$

Where b_0 is the slope, X is the number of concurrent access sessions and b_1 is the intercept. For example, DACIoT can re-evaluate 100 concurrent access sessions against an entirely new access policy (i.e., 1500 access rules) in 205.86 milliseconds using the Hybrid strategy. These numbers show that DACIoT can efficiently adapt to frequent changes in access control policies and maintain authorized access and usage of protected resources in highly dynamic and large-scale IoT environments.

Table 4.3: Estimated Re-evaluation Time of DACIoT Strategies

Strategy	Slope	Intercept	Estimated Time(ms) 1500 rules/100 sessions
Re-evaluate and Decide	1.68	17.11	186
Stop and Re-evaluate	2.28	18.00	246.36
Hybrid	1.86	19.59	205.86

4.4.3 Access Behavior Classification Accuracy

To evaluate the classification accuracy of DACIoT, we used a real-life dataset that consists of 180,000 access logs to the doors of the School of Computing at Queen's University. The data collected over the period from December 1, 2016, to April 30, 2019. Each access log contains the following parameters: user ID (i.e., key ID), door ID, user attributes, door attributes, access time and access value (i.e., decision). We defined the following cases as abnormal access behaviors: (1) If the user is denied access three times with normal access context (i.e., weekdays during morning, afternoon, and evening); (2) If the user is denied access two times in a row in sensitive context settings (i.e., weekends all day, weekdays during the night).

Our classification problem belongs to the Many-to-One category of sequence classification problems. We want our classifier to take a sequence of multiple access requests as input and map it to one behavioral class as output. We implement the DACIoT classifier component using a Recurrent Neural Network (RNN) [113]. We chose RNN because it is designed to work with sequence prediction problems. However, we want the classifier to classify the access behavior every time a new access denial is recorded. Therefore, the length of the input sequence is variable in our case, which requires a careful engineering of the RNN input layer.

To show the effect of data size on the classification accuracy of different classifiers. We also implemented the classifier component using the standard Random Forest classifier (RF) [114]. We used the Python libraries Scikit-learn [115] and Keras [116] to implement the RF and RNN classifiers. For the RNN approach, we used the Long Short-Term Memory (LSTM) [117] network. LSTM overcomes the training problem (i.e., vanishing gradients) associated with the RNNs and in turn has been used in a wide range of applications that involve large datasets.

Preparation of Classification Data

To prepare the classification data, we performed the following steps: First, we selected the top five users based on the total number of accesses, this cut down the data size to 13,296 access logs. Since all parameters of the access logs are categorical and do not provide quantitative information, we applied the one hot encoder (i.e., binarization) to these parameters converting them into numerical input features to train the classifiers.

Second step, we defined 28 input variables as follows: five variables to represent the users' IDs, three variables to represent the user roles, two variables for the day of access, four variables to represent the time of access, 13 variables to represent the doors' IDs, which also represent the doors' locations, and one variable for the access value. We added an output label "Abnormal/Normal" behavior to the data set. The default value to this label is 0 (normal) to all access logs. We set the value to 1 (abnormal) for access logs that contain abnormal access behaviors. Otherwise, the output label is always set to 0 to denote normal access.

Third step, we split the data set into training and testing sets with the percentages 80 and 20, respectively. For the RF classifier, we fed the training data without modifications. For the LSTM classifier, however, we group the access logs between each subsequent abnormal behaviors into one access sequence. The resultant access sequences have different lengths; therefore, we calculate the length of the longest sequence and use zero padding to complete the short sequences.

Finally, we generated 150 synthetic access logs with a different probability distribution (e.g., uniform) from the original data, and evaluated the two classifiers using the mock data.

To show the effect of data size on the classification accuracy of different classifiers, we conducted two experiments using two different data set sizes: 6,000 and 10,000 access logs.

Classification Results and Discussion

One common approach to classification accuracy is to calculate the Area Under the Receiver Operating Characteristic (ROC) curve, abbreviated to AUC. AUC is a measure of how well a classifier can differentiate between two examination samples (e.g., normal/abnormal access behaviors). We chose the AUC as our evaluation metric for the DACIoT classifier because AUC does not depend on the class distribution, which makes it useful for evaluating classifiers predicting rare events such as abnormal access behavior as in our case. In contrast, evaluating the performance using the simple accuracy metric (the number of correct predictions made divided by the total number of predictions made) would favor classifiers that always predict a negative outcome (i.e., normal access behaviors) over rare positive outcome (i.e., abnormal access behaviors). Figure 4.3 shows the performance of the two classifiers when trained and tested on a relatively small number of access logs (4800 logs). The two classifiers score the same AUC over variable values of decision thresholds. The results show that both classifiers poorly distinguish one normal behavior from one abnormal behavior.

Figure 4.4 shows the performance of the two classifiers when validated using the mock data. The RF outperforms the LSTM because there is less chance for short training data to contain long access sequences.

Figure 4.5 shows the performance of the two classifiers when trained and tested

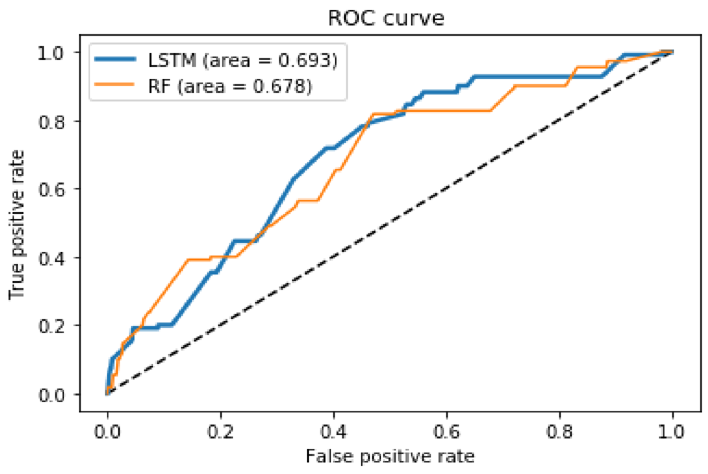


Figure 4.3: Performance of LSTM VS RF, Training Data = 4800 access logs.

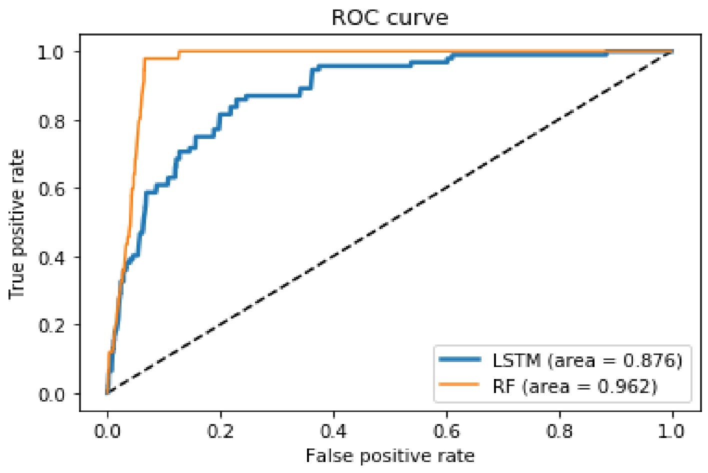


Figure 4.4: The performance of LSTM VS RF, Validation Data = 150 logs.

on relatively large number of access logs. Again, the two classifiers score comparable AUCs. However, The AUC results show that both classifiers can distinguish the two access behaviors with a high level of accuracy. The LSTM outperforms the RF because of the increased chance for the LSTM classifier to learn from longer access sequences in larger training data.

Figure 4.6 shows the performance of the two classifiers when applied to the mock

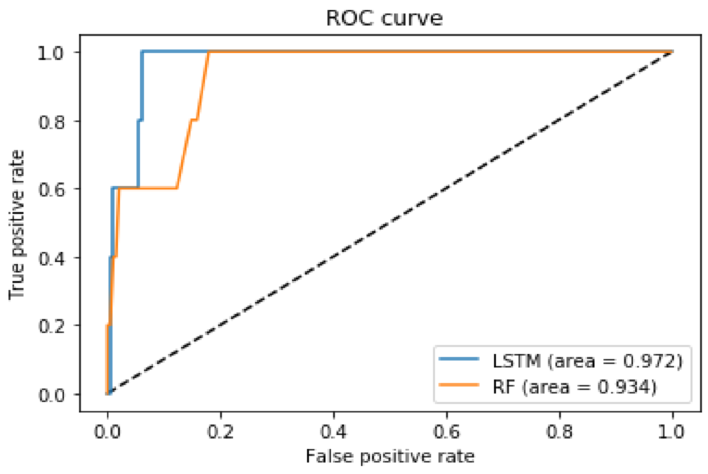


Figure 4.5: The performance of LSTM VS RF, Training Data = 8000 logs.

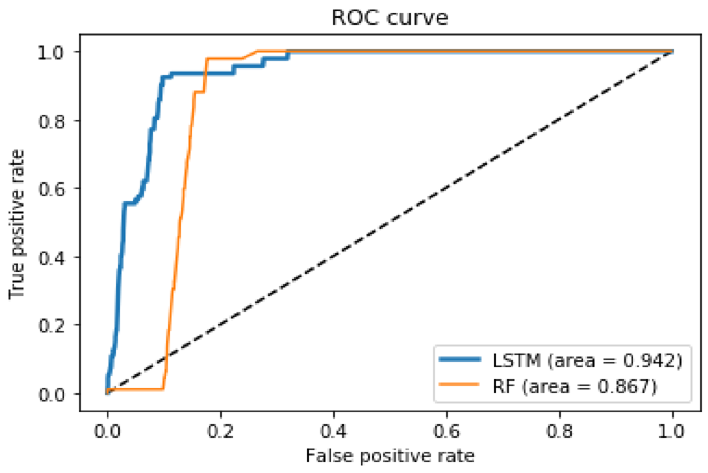


Figure 4.6: The performance of LSTM VS RF, Validation Data = 150 logs.

data. The AUC scores indicate that the LSTM accuracy improves faster on larger training data sets.

4.4.4 Policy Adjustment Accuracy

To measure the added security value that DACIoT brings when adjusting an access control policy, we calculate Policy Adjustment Accuracy (PAA). PAA is the percentage of improvement in the protection of system resources that DACIoT policy adjustment achieves over the original access control policy defined by a system administrator. Assume R is the number of policy adjustments DACIoT recommends and F is the number of policy adjustments the policy administrator refuses. Then the PAA is calculated as follows:

$$PAA = \frac{R - F}{R} \tag{4.4}$$

It follows from the definition of PAA that the added security value of DACIoT is directly proportional to the number of policy adjustments approved by the policy administrator. In this experiment, we run the LSTM classifier on the entire dataset with a total number of 458 users. The classifier reports 466 abnormal access sequences out of a total of 495. We set the adjustment threshold to the minimum value (0.024), which is the maximum allowable number of access denials for a user (three in our case) divided by the maximum sequence length (124 access logs). With this threshold, we allow the adjustment component of DACIoT to generate all possible policy adjustments. There are two types of policy adjustments that DACIoT generates: single and combined attributes. The former is a recommendation for the policy administrator to delete primitive facts that contain a certain attribute (e.g., either subject, object, operation, or environmental), while the latter is a recommendation for policy administrator to delete primitive facts based on two or more attributes (e.g., subject and object).

DACIoT recommended a total of 716 adjustments: 279 adjustments based on the user ID, 301 adjustments based on door ID, 21 adjustments for users on specific doors and 115 adjustments for users during a specific time of the day distributed as follows: 16, 27, 40, 32 adjustments for night, morning, afternoon, and evening times respectively. We present the list of adjustments to the policy administrator for analysis. The administrator approved 68 adjustments and provided the following feedback points:

- Out of 279 (User ID) adjustments, 30 were approved. Out of the 30 adjustments, there are two cases where users should have been revoked their access keys. The abnormal behaviors associated to these two cases are that in the first case, the user committed 79 access denials on the same door, and in the second case, the user committed 10 access denials on 7 different doors. The other 28 (User ID) adjustments are also recommended as combined attributes adjustments: 10 (User ID & Door ID) and 18 (User ID & Time) adjustments.
- Out of the 301 (Door ID) adjustments, only 10 were approved.
- Out of 21 (User ID & Door ID) adjustments, 10 were approved. In the 10 cases, users were blocked access to doors they had been authorized to access, due to administrative changes in the access policy access was denied.
- Out of 115 (User ID & Time), only 18 adjustments were approved. Out of the accepted adjustments, there were 15 that recommend restricting access on afternoon times, three on night times. One possible justification is that there is an increased possibility that employees left their offices for lunch around noontime, which increases the chances for accidental or intentional use of wrong doors by authorized users.

Table 4.4: Policy Adjustments.

	UserID	DoorID	UserID&DoorID	UserID &Time	Total
R	279	301	21	115	716
F	249	291	11	97	648
PAA	0.042 %	0.014 %	0.014 %	0.025 %	0.094%

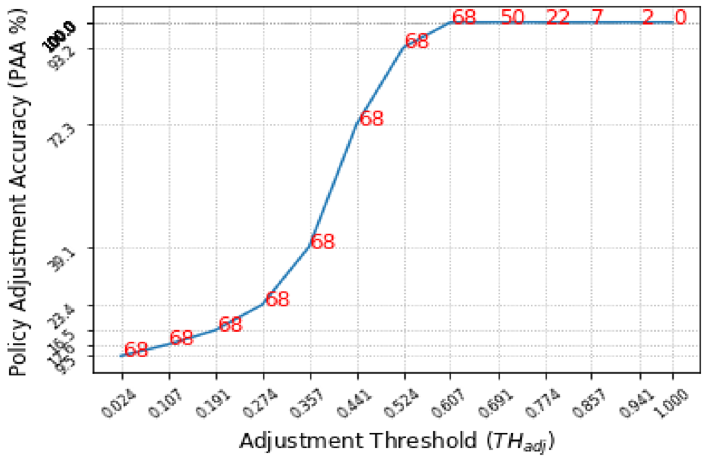


Figure 4.7: Policy adjustment accuracy VS adjustment threshold.

Table 4.4 summarizes DACIoT policy adjustment results. For each type of adjustment, we present the number of generated and refused policy adjustments. The bottom row in the table shows the PAA for each type of adjustment; the percentages represent the contribution of each adjustment type to the overall added security value by DACIoT to the school's resources. Setting the adjustment threshold to a low value in this experiment, resulted in less approved adjustments relative to the overall adjustments, thus the low value of overall PAA was 0.094%.

Figure 4.7 shows the DACIoT policy adjustment accuracy versus variable values of the adjustment threshold. The results show that DACIoT starts with low PAA at small values of TH_{adj}. As TH_{adj} increases, the PAA improves significantly and the total number of approved policy adjustments (plotted in the figure) remains constant. When TH_{adj} equals 0.607 the PAA reaches 100%. At this value of TH_{adj}, all policy adjustments recommended by DACIoT are approved by the policy administrator. When TH_{adj} exceeds 0.607, PAA remains constant; however, the total number of recommended policy adjustments starts to decrease until it reaches zero recommendations for TH_{adj} values 0.941 to 100. Therefore, 0.607 is the optimum adjustment threshold value in this use-case, where DACIoT is capable of striking a balance between the policy adjustment accuracy and the total number of approved policy adjustments.

4.5 Summary

Access control is a key enabler for the adoption of IoT technology. The proliferation of IoT devices in our surroundings, collecting and disseminating sensitive information raises considerable security and privacy concerns. The highly dynamic nature of the IoT environment poses unique access control challenges rendering standard access control policies, models, and mechanisms unsuitable for IoT scenarios. In this chapter, we identified the limitations of current access control approaches and the requirements of a robust and reliable access control mechanism that can efficiently control access to IoT devices and dynamically adapt to frequent changes in access contexts.

Our proposed DACIoT is a dynamic access control framework for IoT environments. DACIoT improves access policy management by enabling the generation and adjustment of conflict-free and context-aware access control rules at runtime. DACIoT ensures continuous authorized access to IoT devices at the time of access request; as well as during the lifetime of an access session taking into consideration dynamic changes in the access control policy and the environmental context in which the access takes place. DACIoT components are presented in comparison to the

XACML access control reference model. A comprehensive description is provided for each component along with the possible approaches from existing technologies that could potentially implement such a component.

The experimental validation of the DACIoT framework demonstrates its ability to generate correct, complete, conflict-free access rules and continuously enforce context-aware and up-to-date access control policies dynamically. Performance evaluation shows that (1) the DACIoT response time rapidly increases as the number of primitive facts increases, however, when the number of primitive facts increases significantly, the response time of DACIoT improves and remains relatively constant for larger number of primitive facts; (2) the re-evaluation time of DACIoT is proportional to the number of concurrent access sessions and applying different re-evaluation strategies affects the rate at which the re-evaluation time is changing with respect to the change in the number of concurrent access sessions. Regardless of which re-evaluation strategy the policy administrator chooses, DACIoT can support reasonable re-evaluation time for time-sensitive applications in resource-limited environments; (3) the accuracy of the access behavior classifier and consequently the accuracy of policy adjustment are inherently data-dependent and subject to the classifier implementation. The RF classifier shows better accuracy on a relatively small data set of training data, while LSTM classifier outperforms the RF classifier on larger training data due to the increased chance for LSTM to learn access patterns from longer access sequences.

Chapter 5

Conclusions and Future Works

The Internet of Things has made a qualitative shift in various areas of human life, such as health care, education, transportation, and natural phenomena surveillance, to name a few. Connecting smart things and devices to the internet has led to the development of smart applications that go beyond exchanging information in the virtual world to include applications that have a physical impact in the real world. The transition from an internet of data to the Internet of Things promises smarter cities, homes, hospitals, and cars. Central to the smartness of the IoT is the context awareness and real-time data access; both have raised the value of the data being collected by IoT devices and led to smarter decision-making and personalized service provisioning. However, the increased sensitivity of the data that is generated by IoT devices raises many security and privacy concerns that impede the widespread adoption of IoT. Controlling access to these devices is the first line of defense against unauthorized usage of users' sensitive information. However, the highly dynamic nature of IoT environments increases the uncertainty about the contexts under which IoT devices can be accessed, which renders the application of standard access control policies and mechanisms to IoT environments infeasible.

5.1 Conclusion

In this thesis, we studied the state-of-the-art access control solutions proposed for IoT from a dynamic authorization perspective. The study provided critical analysis and classification of 26 access control proposals that focus on improving different aspects of traditional access control solutions to meet the security and privacy requirements of IoT environments. Study results reveal that the proposed access control solutions for IoT lack the capacity to address three limitations in traditional access control, including the manual access policy specification, discontinuous access policy enforcement, and over-provisioning of access permissions.

The thesis introduces novel access control concepts and mechanisms that change the way access control policies are specified, enforced and maintained: automatic policy specification, continuous policy enforcement and adaptive policy adjustment. With these mechanisms, we achieve improved security and dynamic adaptability.

Empowered with automatic policy specification schema, our access control approach simplifies the definition and management of access policy for device administrators who lack the technical knowledge and skills to specify access control policies that enforce their security requirements and privacy preferences. The schema enables inexperienced policy administrators to define ad hoc access rules and carries the responsibility of generating correct and conflict-free access control policies at runtime.

The continuous policy enforcement mechanism offers improved security in highly dynamic IoT environments. The policy administrator need not stop and restart the system for changes in access policies to take effect. The mechanism supports instant enforcement of the updated access policies and provides the policy administrator with three strategies to re-evaluate ongoing access sessions: Re-evaluate and Decide, stop and Re-evaluate, and Hybrid. The re-evaluation strategies cater to various security and delay sensitivity requirements of different business applications. Re-valuation and Decide strategy gives priority to resource utilization at the cost of resource security. The Stop and Re-evaluate strategy gives preference to improving the resource security and sacrifices the resource utilization. The Hybrid strategy is a trade-off between security improvement and acceptable latency for the benefit of improving the adaptability of access control while preventing unauthorized access to system resources.

The adaptive policy adjustment mechanism is the last piece of the puzzle. The mechanism classifies user's access behaviors based on current access context and user's access history, and translates the knowledge it acquires from abnormal access behaviors into restrictive policy adjustments; pruning out access rules that assign the user excessive access privileges on system resources. Policy adjustments are defined on the attribute of the user, resource, operation or context that contributes most to the abnormal behavior, and provided as recommendations to the policy administrator. Performance evaluation shows high accuracy in the access behaviors classification and policy adjustments.

5.2 Future Directions

The future work considers the following enhancements to the policy specification, enforcement and adjustment mechanisms:

1. As context information represents the knowledge based on which our access control framework generates access policies, enforces access decisions and classifies

user's behavior. it is crucial to model this knowledge in a natural and formal language such that it can be located, interpreted and shared automatically among different framework components. In the automatic policy specification process, we compute the common guard context under which primitive facts can associate and generate dynamic rules. For measurable context concepts such as time, the common guard context can be easily obtained by finding the maximum common subset of time intervals associated with the matching primitive facts. However, for non-measurable context concepts such as location hierarchies and relationship among users, obtaining the common guard context is challenging.

Interpretation of these contexts requires a knowledge model that defines basic concepts and relationships, as well as the rules that regulate the combination of these concepts. Semantic ontology can be useful for our research work; guard contexts can be modelled in the form of classes, objects, and relations. Ontology can simplify the definition, enforcement, and adjustment of access control policies by following the relationships between classes and objects in the access control process. In addition, our adaptive policy adjustment technique can incorporate complex context concepts to make effective inferences about the access behaviors and derive expressive access control rules and accurate policy adjustments.

2. Our access control framework supports three strategies for the re-evaluation of ongoing access sessions against runtime changes in the access control policies. The Re-evaluate and Decide strategy re-evaluates access sessions and suspends the prevented sessions. The Stop and Re-evaluate strategy stops all access sessions and restores the permitted ones. The Hybrid strategy switches between

re-evaluation strategies based on application-dependent risk threshold. In the performance evaluation, we assumed the probability for the system to shift into a risky state is normally distributed over time. Results show that the Hybrid strategy makes a fair compromise between system security and resource utilization. An interesting extension is to consider dynamic context factors such as the user access history, resource sensitivity, among others, and use machine learning techniques to switch to determine the appropriate re-evaluation strategy. For example, in high-risk access contexts, we suspend active sessions, while in medium or low-risk access contexts, we keep access sessions active until session re-evaluation takes up to a certain time threshold. Otherwise, active sessions are always suspended.

3. The adaptive policy adjustment technique classifies user's access behaviors into normal and abnormal, and restrictively adjusts access control policies as a countermeasure to prevent abnormal access behaviors in the future. However, some changes in context may result in assigning less access privileges that actually are needed by the user in the current access context and blocking legitimate accesses. Detecting such a change in normal access behavior, and escalating access permissions accordingly are challenging and require further investigation. In the future, we plan to extend our work with an anomaly detection component to detect changes in normal access behaviors (i.e., unseen behaviors) at runtime to maintain accurate and up-to-date access control policies.

Bibliography

[1] K. Ashton, “That ‘internet of things’ thing,” *RFID journal*, vol. 22, no. 7, pp. 97-114, 2009.

[2] M. Roberto, B. Abyi, and R. Domenico, “Define IoT - IEEE Internet of Things,” https://iot.ieee.org/definition.html, may 2015, Accessed Jan. 31st, 2019.

[3] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” *Computer networks*, vol. 54, no. 15, pp. 2787-2805, 2010.

[4] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, 2013.

[5] N. Amy, “Popular Internet Of Things Forecast Of 50 Billion Devices By 2020 Is Outdated IEEE Spectrum - IEEE Spectrum,” https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated., Accessed Aug. 27th, 2019.

[6] National Highway Traffic Safety Administration, “Intelligent Transportation Systems - Connected Vehicle,” https://www.its.dot.gov/research_areas/WhitePaper_connected_vehicle., Accessed Jul. 24th, 2018.

[7] B. Janina, "The top 10 iot application areas - based on real iot projects," https://iot-analytics.com/top-10-iot-project-application-areas-q3-2016, Accessed Dec. 4th, 2016.

[8] H. Ghayvat, S. Mukhopadhyay, X. Gui, and N. Suryadevara, "WSN- and IOT-Based Smart Homes and Their Extension to Smart Buildings," *Sensors (Basel, Switzerland)*, vol. 15, no. 5, pp. 10 350-10 379, 2015.

[9] G. Anthes, "Data brokers are watching you." *Commun. ACM*, vol. 58, no. 1, pp. 28-30, 2015.

[10] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266-2279, 2013.

[11] R. John and H. Simon, "The internet toaster," https://www.livinginternet.com/i/ia_myths_toast.htm, Accessed Feb. 13th, 2019.

[12] S. Ornes, "Core Concept: The Internet of Things and the explosion of interconnectivity," *Proceedings of the National Academy of Sciences*, vol. 113, no. 40, pp. 11 059-11 060, 2016.

[13] "Trojan Room Coffee Pot Biography," https://www.cl.cam.ac.uk/coffee/qsf/coffee.html, Accessed Jan. 2nd, 2019.

[14] Y. Laiwu, C. Deyun, and Z. Hongwei, "Research on key technology of internet of things based on frid," in *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, 2011, pp. 355-358.

[15] H. J. Kim, W. J. Song, and S. H. Kim, “Light-weighted Internet protocol version 6 for low-power wireless personal area networks,” in *2008 IEEE International Symposium on Consumer Electronics*, 2008, pp. 1-4.

[16] N. Nikolov, “Research of the Communication Protocols between the IoT Embedded System and the Cloud Structure,” in *2018 IEEE XXVII International Scientific Conference Electronics - ET*, 2018, pp. 1-4.

[17] M. L. Kome, F. Cuppens, N. Cuppens-Boulahia, and V. Frey, “CoAP Enhancement for a Better IoT Centric Protocol: CoAP 2.0,” in *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, 2018, pp. 139-146.

[18] CISCO, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021 White Paper,” https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, Accessed Apr. 5th, 2019.

[19] T. Danova, “Morgan Stanley: 75 Billion Devices Will Be Connected To The Internet Of Things By 2020,” https://www.businessinsider.com/75-billion-devices-will-be-connected-to-the-internet-by-2020-2013-10, Accessed Aug. 27th, 2019.

[20] HUAWEI, “Internet of Things: Sensing Our Way into the Future,” https://www.huawei.com/ca/industry-insights/technology/digital-transformation/iot, 2018, Accessed Jan. 31st, 2019.

[21] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, "Unlocking the potential of the Internet of Things | McKinsey," https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world, 2018, Accessed Jan. 31st, 2019.

[22] R. Baguley, "Appliance Science: The Internet of Toasters (and other things)," https://www.cnet.com/news/appliance-science-the-internet-of-toasters-and-other-things/, 2015, Accessed Jan. 31st, 2019.

[23] M. Ved, "IoT location-based service technologies at a glance," https://medium.com/sciforce/iot-location-based-service-technologies-at-a-glance-a9adfd1d7754, Accessed Oct. 31st, 2019.

[24] Federal Trade Commission, "FTC Internet of Things Report Outlines Privacy and Security Recommendations for Industry," https://www.insideprivacy.com/united-states/federal-trade-commission/ftc-internet-of-things-report-outlines-privacy-and-security-recommendations-for-industry/, 2015, Accessed Nov. 6th, 2015.

[25] G. Black, *Publicity Rights and Image.* UK: Hart Publishing, 2011, pp. 61-62.

[26] J. Groopman and S. Etlinger, "Consumer perceptions of privacy in the internet of things: What brands can learn from a concerned citizenry," *Altimeter Group: San Francisco, CA, USA*, pp. 1-25, 2015.

[27] Z. Yan and C. Prehofer, "Autonomic Trust Management for a Component-Based Software System," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 6, pp. 810-823, 2011.

[28] V. Suryani, Selo, and Widyawan, "A survey on trust in Internet of Things," in *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2016, pp. 1-6.

[29] M. Blaze, J. Feigenbaum, and A. D. Keromytis, "Keynote: Trust management for public-key infrastructures," in *International Workshop on Security Protocols*. Springer, 1998, pp. 59-63.

[30] K. E. Seamons, T. Chan, E. Child, M. Halcrow, A. Hess, J. Holt, J. Jacobson, R. Jarvis, A. Patty, B. Smith, T. Sundelin, and Lina Yu, "Trustbuilder: negotiating trust in dynamic coalitions," in *Proceedings DARPA Information Survivability Conference and Exposition*, vol. 2, 2003, pp. 49–51.

[31] N. R. Council, *Computers at Risk: Safe Computing in the Information Age*. The National Academies Press, Washington, DC, 1991, p. 83.

[32] A. Jøsang, "A Consistent Definition of Authorization," in *Security and Trust Management*, ser. Lecture Notes in Computer Science, G. Livraga and C. Mitchell, Eds. Springer International Publishing, 2017, pp. 134-144.

[33] P. Samarati and S. C. de Vimercati, "Access Control: Policies, Models, and Mechanisms," in *Foundations of Security Analysis and Design*, ser. Lecture Notes in Computer Science, R. Focardi and R. Gorrieri, Eds. Springer Berlin Heidelberg, 2001, pp. 137-196.

[34] H. C. Assistance, “Summary of the hipaa privacy rule,” *Office for Civil Rights*, 2003, Accessed Oct. 31st, 2019.

[35] R. Sandhu, D. Ferraiolo, and R. Kuhn, “The NIST model for role-based access control: towards a unified standard,” in *ACM workshop on Role-based access control*, vol. 10, 2000, pp. 47-63.

[36] R. Sandhu and P. Samarati, “Access control: principle and practice,” *IEEE Communications Magazine*, vol. 32, no. 9, pp. 40-48, Sep. 1994.

[37] L. E. Holmquist, J. Redström, and P. Ljungstrand, “Token-Based Access to Digital Information,” in *Handheld and Ubiquitous Computing*, ser. Lecture Notes in Computer Science, H.-W. Gellersen, Ed. Springer Berlin Heidelberg, 1999, pp. 234-245.

[38] L. Gong, “A secure identity-based capability system,” in *Proceedings. 1989 IEEE Symposium on Security and Privacy*, 1989, pp. 56-63.

[39] D. E. Bell and L. J. LaPadula, “Secure Computer Systems: Mathematical Foundations,” https://apps.dtic.mil/docs/citations/AD0770768, MITRE CORP BEDFORD MA, Tech. Rep. MTR-2547-VOL-1, 1973, Accessed Oct. 10th, 2015.

[40] D. Elliott Bell, “Bellla padula model,” *Encyclopedia of cryptography and security*, pp. 74-79, 2011.

[41] K. J. Biba, “Integrity Considerations for Secure Computer Systems,” https://apps.dtic.mil/docs/citations/ADA039324, MITRE CORP BEDFORD MA, Tech. Rep. MTR-3153-REV-1, Aprl 1977, Accessed Nov. 3rd, 2016.

[42] D. Chappell, "Introducing the Windows Azure Platform," pp. 3-21, 2009.

[43] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone *et al.*, "Guide to attribute based access control (abac) definition and considerations (draft)," *NIST special publication*, vol. 800, no. 162, 2013.

[44] V. C. Hu, D. R. Kuhn, D. Ferraiolo, and J. Voas, "Attribute-Based Access Control," *Computer*, vol. 48, no. 2, pp. 85-88, 2015.

[45] M. Gupta, F. Patwa, and R. Sandhu, "An Attribute-Based Access Control Model for Secure Big Data Processing in Hadoop Ecosystem," in *Proceedings of the Third ACM Workshop on Attribute-Based Access Control.* ACM, 2018, pp. 13-24.

[46] T. White, *Hadoop: The definitive guide.* " O'Reilly Media, Inc.", 2012.

[47] "Apache Ranger - Introduction," https://ranger.apache.org/, 2019, Accessed Aug. 7th, 2019.

[48] "Apache Sentry," https://sentry.apache.org/, 2016, Accessed Aug. 7th, 2019.

[49] J. Park and R. Sandhu, "The UCONABC Usage Control Model," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 128-174, 2004.

[50] J. Park, R. Sandhu, and J. Schifalacqua, "Security architectures for controlled digital information dissemination," in *Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00)*, 2000, pp. 224-233.

[51] S. Godik and T. Moses, “Oasis extensible access control markup language (xacml),” *OASIS Committee Secification cs-xacml-specification-1.0*, 2002.

[52] S. Saklikar and S. Saha, “Next Steps for Security Assertion Markup Language (Saml),” in *Proceedings of the 2007 ACM Workshop on Secure Web Services*. ACM, 2007, pp. 52-65.

[53] E. Rescorla and A. Schiffman, “The Secure HyperText Transfer Protocol,” http://www.rfc-editor.org/info/rfc2660, 1999, Accessed Oct. 31st, 2019.

[54] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, “Foundations of JSON Schema,” in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 263-273.

[55] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” http://www.rfc-editor.org/info/rfc7252, 2014, Accessed Oct. 31st, 2019.

[56] X. Ma and W. Luo, “The Analysis of 6lowpan Technology,” in *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, vol. 1, 2008, pp. 963-966.

[57] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren, *Handbook of elliptic and hyperelliptic curve cryptography*. Chapman and Hall/CRC, 2005.

[58] A. Katok and B. Hasselblatt, *Introduction to the modern theory of dynamical systems*, 1st ed. Cambridge university press, 1995, vol. 54, pp. 1-4.

[59] D. E. Seborg, D. A. Mellichamp, T. F. Edgar, and F. J. D. III, *Process Dynamics and Control*, 3rd ed. John Wiley & Sons, 2010.

[60] M. J. Moyer and M. Abamad, "Generalized role-based access control," in *Proceedings 21st International Conference on Distributed Computing Systems*, 2001, pp. 391-398.

[61] P. N. Mahalle, B. Anggorojati, N. R. Prasad, R. Prasad *et al.*, "Identity authentication and capability based access control (iacac) for the internet of things," *Journal of Cyber Security and Mobility*, vol. 1, no. 4, pp. 309-348, 2013.

[62] B. Anggorojati, P. N. Mahalle, N. R. Prasad, and R. Prasad, "Capability-based access control delegation model on the federated iot network," in *Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on.* IEEE, 2012, pp. 604-608.

[63] J. Jindou, Q. Xiaofeng, and C. Cheng, "Access Control Method for Web of Things Based on Role and SNS," in *2012 IEEE 12th International Conference on Computer and Information Technology*, 2012, pp. 316-321.

[64] G. Zhang and J. Tian, "An extended role based access control model for the Internet of Things," in *2010 International Conference on Information, Networking and Automation (ICINA)*, vol. 1, 2010, pp. V1-319-V1-323.

[65] A. S. M. Kayes, J. Han, and A. Colman, "A Semantic Policy Framework for Context-Aware Access Control Applications," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2013, pp. 753-762.

[66] B. Sean, H. Frank van, H. Jim, H. Ian, M. Deborah, P. S. Peter, and A. S. Lynn, “OWL - Semantic Web Standards,” https://www.w3.org/OWL/, 2012, Accessed Feb. 1st, 2017.

[67] H. Ian, P. S. Peter, B. Harold, T. Said, G. Benjamin, and D. Mike, “SWRL: A Semantic Web Rule Language Combining OWL and RuleML,” http://www.daml.org/swrl/proposal/rules-all.html, 2004, Accessed Feb. 1st, 2017.

[68] J. L. Hernåndez-Ramos, A. J. Jara, L. Marín, and A. F. S. Gómez, “DCapBAC: embedding authorization logic into smart things through ECC optimizations,” *International Journal of Computer Mathematics*, vol. 93, no. 2, pp. 345-366, 2016.

[69] D. Johnson, A. Menezes, and S. Vanstone, “The Elliptic Curve Digital Signature Algorithm (ECDSA),” *International Journal of Information Security*, vol. 1, no. 1, pp. 36-63, 2001.

[70] J. L. Hernandez-Ramos, M. P. Pawlowski, A. J. Jara, A. F. Skarmeta, and L. Ladid, “Toward a lightweight authentication and authorization framework for smart objects,” *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 4, pp. 690-702, 2015.

[71] M. Bafandehkar, S. M. Yasin, R. Mahmod, and Z. M. Hanapi, “Comparison of ECC and RSA Algorithm in Resource Constrained Devices,” in *2013 International Conference on IT Convergence and Security (ICITCS)*, 2013, pp. 1-3.

[72] N. Ye, Y. Zhu, R.-c. Wang, R. Malekian, and L. Qiao-Min, "An efficient authentication and access control scheme for perception layer of internet of things," *Applied Mathematics & Information Sciences*, vol. 8, no. 4, p. 1617, 2014.

[73] E. Barka, S. S. Mathew, and Y. Atif, "Securing the Web of Things with Role-Based Access Control," in *Codes, Cryptology, and Information Security*, ser. Lecture Notes in Computer Science, S. El Hajji, A. Nitaj, C. Carlet, and E. M. Souidi, Eds. Springer International Publishing, 2015, pp. 14-26.

[74] D. Fensel, "Ontologies," in *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, D. Fensel, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 11-18.

[75] ISO, "Information technology - Open Systems Interconnection - Security frameworks for open systems: Access control framework - Part 3," https://www.iso.org/obp/ui/iso:std:iso-iec:10181:-3:ed-1:v1:en, Accessed Oct. 31st, 2019.

[76] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509-517, 1975.

[77] H. F. Atlam, A. Alenezi, R. J. Walters, G. B. Wills, and J. Daniel, "Developing an Adaptive Risk-Based Access Control Model for the Internet of Things," in *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2017, pp. 655-661.

[78] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, 2017, pp. 557-564.

[79] H. F. Atlam, M. O. Alassafi, A. Alenezi, R. J. Walters, and G. B. Wills, "Xacml for building access control policies in internet of things." in *IoTBDS*, 2018, pp. 253-260.

[80] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based Encryption for Fine-grained Access Control of Encrypted Data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security.* ACM, 2006, pp. 89-98.

[81] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," in *2007 IEEE Symposium on Security and Privacy (SP '07)*, 2007, pp. 321-334.

[82] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance evaluation of Attribute-Based Encryption: Toward data privacy in the IoT," in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 725-730.

[83] Z. Guoping and G. Wentao, "The research of access control based on ucon in the internet of things," *Journal of Software*, vol. 6, no. 4, pp. 724-731, 2011.

[84] D. Hussein, E. Bertin, and V. Frey, "A Community-Driven Access Control Approach in Distributed IoT Environments," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 146-153, 2017.

[85] S. Gusmeroli, S. Piccione, and D. Rotondi, "A capability-based security approach to manage access control in the Internet of Things," *Mathematical and Computer Modelling*, vol. 58, no. 5, pp. 1189-1205, 2013.

[86] J. Liu, Y. Xiao, and C. P. Chen, "Authentication and access control in the internet of things," in *2012 32nd International Conference on Distributed Computing Systems Workshops*. IEEE, 2012, pp. 588–592.

[87] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, "Openid connect core 1.0 incorporating errata set 1," http://openid.net/specs/openid-connect-core-1_0.html., 2014, Accessed Nov. 11th, 2017.

[88] S. Sciancalepore, M. Pilc, S. Schröder, G. Bianchi, G. Boggia, M. Pawłowski, G. Piro, M. Płóciennik, and H. Weisgrab, "Attribute-Based Access Control Scheme in Federated IoT Platforms," in *Interoperability and Open-Source Solutions for the Internet of Things*, I. Podnar Žarko, A. Broering, S. Soursos, and M. Serrano, Eds. Springer International Publishing, 2017, vol. 10218, pp. 123-138.

[89] A. A. E. Kalam, S. Benferhat, A. Miège, R. E. Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and G. Trouessin, "Organization Based Access Control," in *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*. IEEE Computer Society, 2003, p. 120.

[90] I. B. Pasquier, A. A. Ouahman, A. A. E. Kalam, and M. O. d. Montfort, "SmartOrBAC security and privacy in the Internet of Things," in *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, 2015, pp. 1-8.

[91] A. A. El Kalam and Y. Deswarte, "Multi-orbac: A new access control model for distributed, heterogeneous and collaborative systems," in *8th IEEE International Symposium on Systems and Information Security*, 2006, p. 1.

[92] A. Abou El Kalam, Y. Deswarte, A. Baïna, and M. Kaâniche, "PolyOrBAC: A security framework for Critical Infrastructures," *International Journal of Critical Infrastructure Protection*, vol. 2, no. 4, pp. 154-169, 2009.

[93] Luis, D. Rivera, Cruz-Piris, I. Marsa-Maestre, E. De la Hoz, and J. R. Velasco, "Access Control Mechanism for IoT Environments Based on Modelling Communication Procedures as Resources," *Sensors*, vol. 18, no. 3, p. 917, 2018.

[94] T. Hardjono, E. Maler, M. Machulak, and D. Catalano, "User-Managed Access (UMA) Profile of OAuth 2.0," https://docs.kantarainitiative.org/uma/rec-uma-core.html, Accessed Jan. 9th, 2018.

[95] S. Zamfir, T. Balan, I. Iliescu, and F. Sandu, "A security analysis on standard IoT protocols," in *2016 International Conference on Applied and Theoretical Electricity (ICATE)*, 2016, pp. 1-6.

[96] E. A. Alkeem, C. Y. Yeun, and M. J. Zemerly, "Security and privacy framework for ubiquitous healthcare IoT devices," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2015, pp. 70–75.

[97] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2017, pp. 618-623.

[98] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart Contract-Based Access Control for the Internet of Things," *IEEE Internet of Things Journal*, p. 1, 2018.

[99] J. Wang, H. Wang, H. Zhang, and N. Cao, "Trust and Attribute-Based Dynamic Access Control Model for Internet of Things," in *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2017, pp. 342-345.

[100] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "FairAccess: a new Blockchain-based access control framework for the Internet of Things," *Security and Communication Networks*, vol. 9, no. 18, pp. 5943-5964, 2016.

[101] P. N. Mahalle, P. A. Thakre, N. R. Prasad, and R. Prasad, "A fuzzy approach to trust based access control in internet of things," in *Wireless VITAE 2013*, 2013, pp. 1-5.

[102] A. X. Liu, F. Chen, J. Hwang, and T. Xie, "Designing Fast and Scalable XACML Policy Evaluation Engines," *IEEE Transactions on Computers*, vol. 60, no. 12, pp. 1802-1817, 2011.

[103] D. Lin, P. Rao, R. Ferrini, E. Bertino, and J. Lobo, "A similarity measure for comparing XACML policies," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 9, pp. 1946-1959, 2012.

[104] E. Shaw, K. G. Ruby, and J. M. Post, "The insider threat to information systems," *Security Awareness Bulletin*, vol. 2, no. 98, pp. 1-10, 1998.

[105] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12-22, 2018.

[106] I. Hafeez, A. Y. Ding, M. Antikainen, and S. Tarkoma, "Toward Secure Edge Networks Taming Device to Device (D2D) Communication in IoT," *arXiv e-prints*, p. arXiv:1712.05958, 2017.

[107] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734-11 753, 2012.

[108] E. Alepis and C. Patsakis, "Hey doc, is this normal?: exploring android permissions in the post marshmallow era," in *International Conference on Security, Privacy, and Applied Cryptography Engineering.* Springer, 2017, pp. 53-73.

[109] CanaKit Corporation, "Raspberry pi 3," https://www.raspberrypi.org /documentation/hardware/raspberrypi/, Accessed May. 4th, 2017.

[110] Texas Instruments, "Simplelink^{TM} bluetooth low energy/multi-standard sensortag," http://www.ti.com/tool/CC2650STK, Accessed May. 12th, 2017.

[111] Belkin, "Belkin's wemo switch," http://www.wemo.com/, Accessed Jun. 8th, 2016.

[112] S. Jajodia, P. Samarati, M. L. Sapino, and V. Subrahmanian, "Flexible support for multiple access control policies," *ACM Transactions on Database Systems (TODS)*, vol. 26, no. 2, pp. 214-260, 2001.

[113] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[114] Tin Kam Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, 1995, pp. 278-282 vol.1.

[115] Scikit Learn, "scikit-learn: machine learning in Python — scikit-learn 0.21.2 documentation," https://scikit-learn.org/stable/, Accessed Jul. 10, 2019.

[116] "Home - Keras Documentation," https://keras.io/, Accessed Jul. 18, 2019.

[117] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735-80, 1997.