# A Lightweight Attribute-Based Access Control System for IoT.

A Proposal Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Samiul Monir

# Abstract

The evolution of the Internet of things (IoT) has made a significant impact on our daily and professional life. Home and office automation are now even easier with the implementation of IoT. Multiple sensors are connected to monitor the production line, or to control an unmanned environment is now a reality. Sensors are now smart enough to sense an environment and also communicate over the Internet. That is why, implementing an IoT system within the production line, hospitals, office space, or at home could be beneficial as a human can interact over the Internet at any time to know the environment. 61% of International Data Corporation (IDC) surveyed organizations are actively pursuing IoT initiatives, and 6.8% of the average IT budgets is also being allocated to IoT initiatives. However, the security risks are still unknown, and 34% of respondents pointed out that data safety is their primary concern [1].

IoT sensors are being open to the users with portable/mobile devices. These mobile devices have enough computational power and make it difficult to track down who is using the data or resources. That is why this research focuses on proposing a dynamic access control system for portable devices in IoT environment. The proposed architecture evaluates user context information from mobile devices and calculates trust value by matching with defined policies to mitigate IoT risks. The cloud application acts as a trust module or gatekeeper that provides the authorization access to READ, WRITE, and control the IoT sensor.

The goal of this thesis is to offer an access control system that is dynamic, flexible, and lightweight. This proposed access control architecture can secure IoT sensors as well as protect sensor data. A prototype of the working model of the cloud, mobile application, and sensors is developed to prove the concept and evaluated against automated generated web requests to measure the response time and performance overhead. The results show that the proposed system requires less interaction time than the state-of-the-art methods.

# Acknowledgements

First of all, I would like to thank my thesis supervisor, Dr. Ralph Deters, for his support, guidance, patience throughout my graduate study. I would like to thank Dr. Julita Vassileva, Dr. Chanchal Roy and Dr. Khan A. Wahid for being my committee member. I also would like to thank other faculty members and staffs at the Department of Computer Science, who has been very helpful throughout my study at the University of Saskatchewan. Finally, I would like to thank my family, friends, and fellow members of the MADMUC lab for their selfless support.

# Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| API | Application Programming Interface |
| CoAP | The Constrained Application Protocol |
| CPU | Central Processing Unit |
| CRUD | Create, Read, Update and Delete |
| DDS | Data Distribution Service |
| DDoS | Distributed Denial of Service |
| ECC | Elliptic Curve Cryptography |
| ESB | Enterprise Server Bus |
| GPS | Global Positioning System |
| GSM | Global System for Mobile Communications |
| GUI | Graphical User Interface |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| ICT | Information Communications Technology |
| IoT | Internet of Things |
| IP | Internet Protocol |
| IT | Information Technology |
| JSON | JavaScript Object Notation |
| KB | Kilo Byte |
| M2M | Machine-to-Machine |
| MAC | Media Access Control |
| MAM | Mobile application management |
| MB | Mega Byte |
| MDM | Mobile device management |
| MMS | Multimedia Messaging Service |
| MQTT | Message Queue Telemetry Transport |
| PDA | Personal Digital Assistant |
| PDP | Policy Decision Point |
| REST | Representational State Transfer |
| RFID | Radio Frequency IDentification |
| RTT | Round Trip Time |
| SDK | Software Development Kit |
| SOAP | Simple Object Access Protocol |
| TCP | The Transmission Control Protocol |
| UDP | User Datagram Protocol |
| URI | Universal Resource Identifier |
| VoIP | Voice Over Internet Protocol |
| WSDL | Web Services Description Language |
| XML | Extensible Markup Language |

# Chapter 1

# Introduction

The Internet of Things (IoT) is commonly used to name a set of connected objects (or things) that are directly connected to each other or connected through a router or cloud service using the Internet [2]. International Telecommunication Union (ITU) first proposed the concept of IoT in 2005. Then the era of IoT begun and the concept of IoT has evolved over time [3, 4]. The core idea is to create a network of connected entities (see Figure 1.1). These entities could be human beings, computers, books, cars, home appliances, Smartphones, etc., and have a locatable and readable address on the Internet. They can communicate by opening a channel with any other entity, providing and receiving services at any time [3].

The primary function of IoT nodes is to collect environment knowledge for the authorized user. These sensor nodes are reliable, portable, cheap and easy to integrate. It also has fewer computational complexities. These technologies are serving the building blocks of automotive, healthcare, logistics, environmental monitoring, and many others. In a centralized approach, the application platform is responsible for gathering information from entities within the network and provide support to other entities. The application platform on the Internet controls the authentication, authorization and information flow [3, 4, 5]. In this thesis, only the centralized approach has been considered.

Mobile phone technology has improved considerably over time [6]. Earlier days' cell phones were used only to make calls and send short text messages. But now, it has become more interactive and advanced with the high internet bandwidth, CPU speed, memory, and storage capacity. These advancements are now encouraging a user to use their mobile devices in their personal and professional environments. It adds flexibility to the user doing their professional and household jobs and helps them to become more productive as they are using their preferred and known technology in their daily life. But as Smartphone is more powerful than IoT sensor nodes, it can track, read and control a sensor node with appropriate authorization. A Smartphone can also participate in the IoT networks, and as it has more processing power compared to other low-powered devices, the security can be at risk if the Smartphone does not have proper authority. While a Smartphone is connecting to the IoT network, there is no way to confirm that the real user is using the device. If the Smartphone gets connected to the patient monitoring system in the hospital or any house security system, the security of that network could be breached.

An adaptive solution to IoT security risks is becoming essential as the IoT devices have a variety of computational complexity and memory. Such a solution can allow users to secure their data, and resources, as well

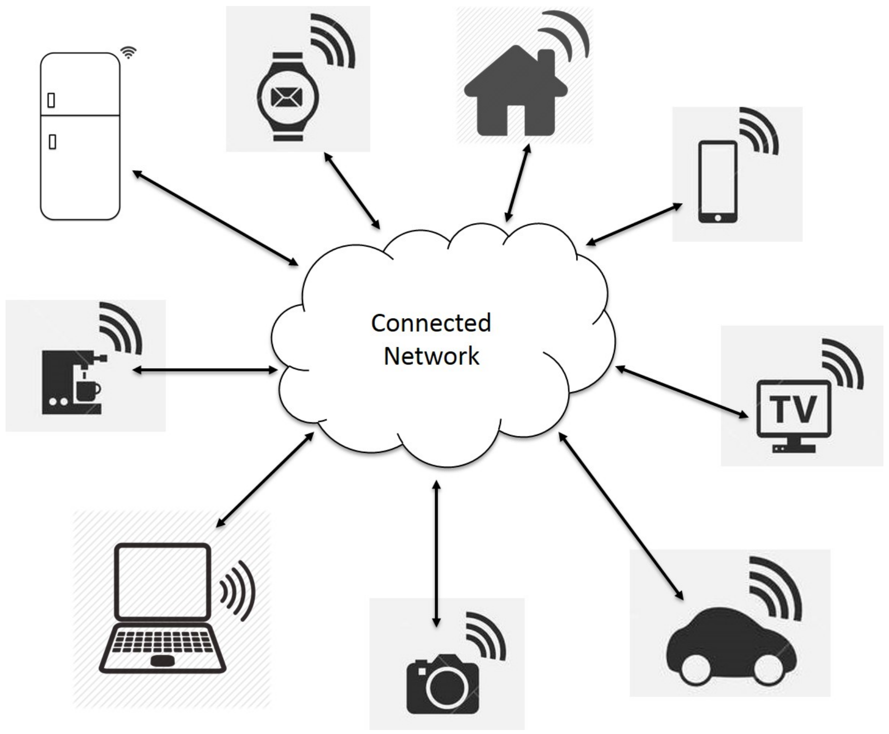


**Figure 1.1:** IoT with Connected Entities

as, they can operate the whole network with their portable devices. A dynamic attribute-based authorization system formed on the device's contextual information is proposed to ensure a secured interaction between user's mobile device and IoT sensor nodes. The proposed approach collects device contextual information and calculates two level trust value based on the policy set by the administrator of the network.

The rest of this paper is organized as follows: Chapter 2 introduces the research questions, challenges, and goals. Chapter 3 discusses previous and related works. Chapter 4 provides explicit insight into the proposed approach. Chapter 5 presents the experiments and discuss the results. Finally, Chapter 6 concludes with the possible contributions and future directions.

# Chapter 2

# Problem Definition

Smart connected devices introduce new opportunities to enable new services and merge new technologies with modern life. Each device is interconnected and can be accessed through the standards of the web. As these devices are individually accessible, any user can join the network through any of the devices and access data from any endpoint, at any time, from any device as shown in figure 2.1.

From the recent studies discussed in Chapter 3 and figure 2.2, it is visible that how the key-based access control process works. A user makes a request to access IoT resources through mobile devices. These devices are called "Things" and these become the end nodes in the IoT environment. These end nodes can communicate and create a new network by themselves. These IoT nodes can access data from other nodes or monitor each other using TCP or UDP data package. At first, each node will register on a nearby trustworthy access point denoted as *Registration Authority or RA*. Nodes are provided with a key that can be shared with the other IoT nodes to have read, write or control privileges. After receiving a request, the recipient node can authenticate this key from RA and validate the access authority of that node.

Users are now dependent on multiple devices that are connected to each other. Not only, these devices have included Smartphones and tablets but also Smartwatch, Smart TV, and other devices with Smart sensing power. Various IoT applications can be build using these devices such as home solutions and patient monitoring system. Although these smart devices are highly constrained in memory and computational power, these devices have the ability to monitor and collect environment data efficiently.

Apart from the benefits of IoTs, security is the main challenge in terms of implementing IoT solutions, and privacy and inappropriate access are the top of the list. As these devices create an open network where any device can join and interact with other devices, maintaining proper authorization becomes necessary.

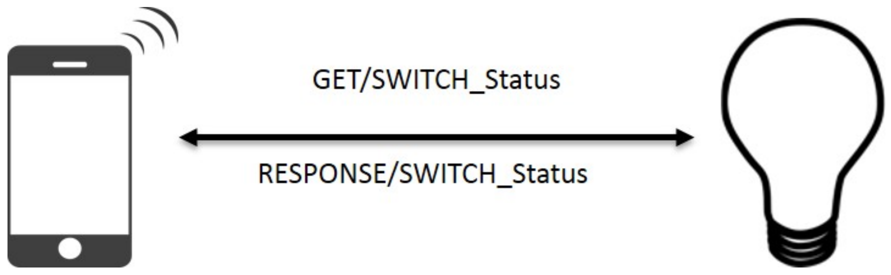


**Figure 2.1:** IoT resource interaction with Mobile.

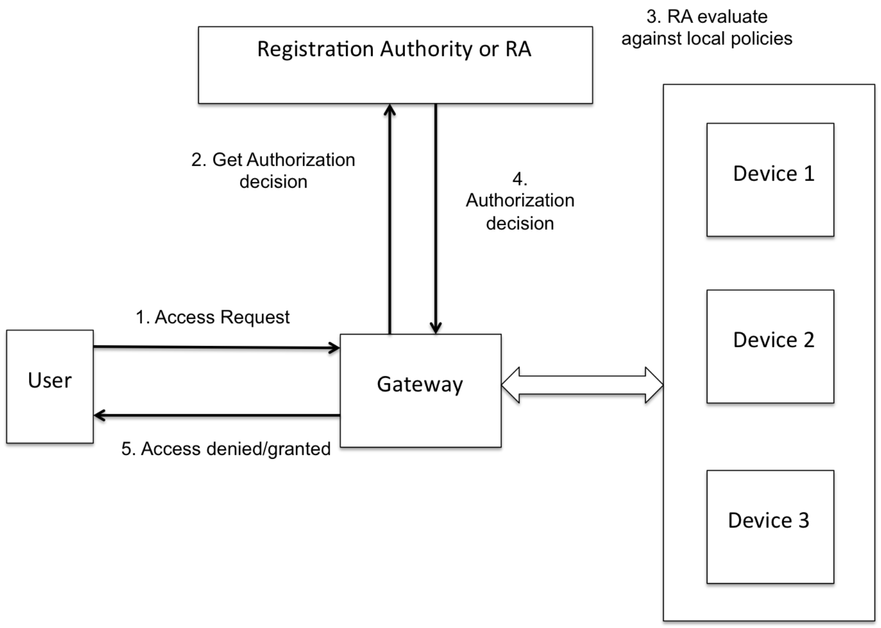

3. RA evaluate against local policies
Registration Authority or RA
2. Get Authorization decision
4. Authorization decision
Device 1
Device 2
Device 3
1. Access Request
User
Gateway
5. Access denied/granted

**Figure 2.2:** Key-based access control in IoT environment. [7]

Traditional network solutions do not align well to develop IoT applications, and if any of the devices are compromised, then the possibility of malicious attacks are also rising and the privacy will become vulnerable. Hacking into cameras, violating privacy, and accessing contents (pictures and personal information) are some of the security threats related to the IoT and can have dangerous outcome [8]. The IoT environment becomes complex, and the privacy issues become complicated with the new technologies. These concerns are rising because of the new network structure, scene, terminal equipment and other factors of IoT and cannot be solved by the traditional firewall or key chain pair or authentication protocols [4].

Therefore, unauthorized access needs to be evaluated properly. As access control technologies are still an essential element to address the security and privacy risks in computer networks, the IoT is still an immature technology and the safety of the IoT especially access control of miniaturized "things" has become the most challenging in aspects of security and privacy [9].

Although IoT has better prospects, few well-known security challenges (DDOS: Denial of Service Attacks on the Internet of Thing [10], Eavesdropping in the Internet of Things [11, 12], Privacy and access control [7, 13, 14], and others) arise due to the introduction of IOT. In this thesis, access control issues are considered. The recent development of IoT leads to a situation where privacy issue is becoming more and more challenging and personal information such as health-related data, personal and official documents, etc. are kept, monitored and controlled by the network of things.

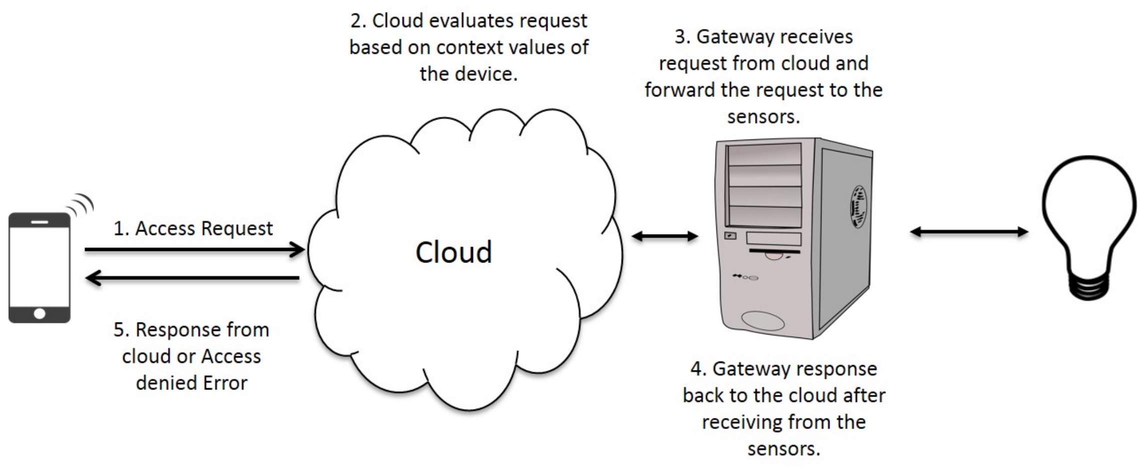

2. Cloud evaluates request based on context values of the device.
3. Gateway receives request from cloud and forward the request to the sensors.
1. Access Request
Cloud
5. Response from cloud or Access denied Error
4. Gateway response back to the cloud after receiving from the sensors.

**Figure 2.3:** A lightweight attribute-based access control system in IoT environment.

## 2.1 Research Goals

The objective of this thesis is to establish a centralized access control system between the connected IoT nodes. To achieve the proposed objective, a lightweight attribute-based access control system is developed which can verify a request made by a Smartphone or any other powerful devices to READ, WRITE or control other IoT nodes. Figure 2.3 illustrates the proposed solution. The proposed access control system also has the following features:

1. Provide a lightweight solution to mitigate the privacy-related security threats associated with IoT [4, 15]. The cloud is responsible to handle heavy tasks where other low powered devices are only performing light tasks.

2. Cost effective solution to overcome the violations of accessing unauthorized contents [5, 8, 9, 15].

3. Provide a scalable solution to protect privacy [9].

In this thesis, the primary research goal is to achieve a trust based on attributes that are collected from the sender device, to provide an access control system implemented in the cloud to verify a request while accessing sensitive IoT sensor resources.

# Chapter 3

# Literature Review

The main objective of this research is to create a dynamic access control model to secure the interaction between mobile and the IoT devices. The access control model considers mobile context information and the defined policies to determine the validity of a request. To the user, the system should be flexible, reliable and trustworthy. Previous inquiries have been conducted related to the access control, web access policies, and recent courses.

This chapter focuses on the availability of security of the interaction of mobile devices and other smart devices with sensors which has less memory and computing power in IoT space. It also provides a better understanding of the interaction of mobile devices and other devices in IoT environment and discusses the problems of having privacy and access control risks. To further establish a bi-directional trust that holds the access control system of mobile customers and IoT devices, the following fields are examined in this thesis: IoT, Access Control, IoT Access Control risks and solutions, Context-based security, Mobile device security, cloud computing, and web services.

## 3.1 IoT in Practice

Internet of things (IoT) is the next biggest revolution in the Automation and IT world. IoT would transform the existing Internet into a fully integrated form of Internet where it would deliver a smarter connectivity among things or objects. With the growth of less powered Internet-enabled embedded systems, the number of IoT applications is expanding rapidly. An Internet-connected weather-monitoring station, an Internet-enabled air conditioner, an Internet-connected car, Smart watches, sensors and actuators to the Internet where the devices are intelligently linked together enabling various ways to communicate are the good examples of IoT applications [16, 17].

Mark [18] argued that the introduction of technology is evolving at a speedy rate. During the past decades, the development and existence of IoT in our lives were silent. However with the rapid development of wireless technologies (RFID, Wi-Fi, 4G, etc.), this technology is offering smart monitoring and control applications in our daily life [19, 20, 21]. The concept of IoT is many-folded as it embraces many different technologies, and services in the ICT market in next ten year [22].

A large number of low-cost sensors and wireless communication bring new demands for the communication technology. According to CISCO, there are approximate 6.3 billion people living on this planet and 500 million devices connected to the Internet, and that means 0.08 devices per person. Now with the rapid expansion of low powered devices and Smartphones, CISCO predicts that by 2020, 50 billion devices will be connected to the Internet, which leads to 6.58 devices per person. The above discussion of this paragraph was inspired from [23].

Evans [23] also mentioned that IoT had reached a certain point where disparate networks must need to integrate and interoperate under common standards. Academia, Government, organization and business people need to work on making the common standards together. He also suggests how IoT can gain acceptance by the wider public by exploring the opportunities of services and delivering applications that add tangible values to the lives of people.

In the era of IoT, low powered machines are getting smarter with computational power and can handle more human tasks. Now smart car now can have the remote starter, engine check option, automated-locking, driving the car with mobile and other essential features. With the advancement of wireless sensor networks, real-time monitoring in healthcare application or an accelerometer for movement attached to the cow in a farm environment is becoming more realistic. Thus, the interaction between humans and machines are becoming complicated and introducing variations. However, humans are required to trust the machines and feel safe [10, 24].

IoT has the capabilities to make homes, cars and our surroundings smarter. These innovative technologies provide assistance to our everyday activities, health support, energy efficiency, security, and comfort. By adding intelligence capabilities to our surroundings such as homes, cars, office, personal health monitoring could provide increased life quality. It is expected that disable and elderly people will be much benefited from this advancement. Most of the research in IoT are focused on wireless technologies that are supportive of remote data controlling, sensing, and transferring as well as integrating RFID, Bluetooth, which have been used to embed intelligence into the environment [25, 26]. In a similar note, Consumers are in thirst of control of our daily life's things and Gaglio and Lo [27] argued that this transformation process is making our every day's things into useful home applications, car applications, and many others that the consumers can control. A lot of applications for smart TV, smart fridge, smart car, network energy monitors, and activity tracking systems now exist in homes as well as at workplaces. These systems will collect detailed data and information about the activities of concerning objects/ persons and share this information to the relevant or authorize personnel.

The concept of IoT is now changing the current world to an entirely ubiquitous one. IoT is now changing our perspective of the Internet radically by including all kinds of electronic devices into the network. Network communication includes things, people and their environment with the help of near-field communications, creating smart objects, embedded devices, the Internet, cloud computing, and GPS-enabled localization. By including all of these physical objects, it becomes easier to understand and the reaction of the environment [28].

Weber [29] describes IoT in the form of an Internet-based global architecture which will enable trade of services and goods in the global chain of supply. However, in the same way, it might have an impact on the privacy and security of all stakeholders. IoT has a significant impact on ubiquitous monitoring. Control, Awareness, context, intrusion, trust, boundaries, and justification are the key factors of ubiquitous monitoring [30].

Data collection, enabling real-time responses, improving access and control of devices, increasing efficiency and productivity, and connecting technologies are the essential features that could benefit our personal life as well as organizations. IoT creates the opportunity to collect data more frequently and allows transmitting information to another device. It provides the opportunity to monitor and optimize outcome in real-time. Another great benefit is the ability to access and control Internet-connected devices. For example, people can control their electronic stove, microwave, smart TV or even check fire alarm system when they are away from their home [31].

IoT gains significant productivity in a variety of areas including healthcare, home automation, equipment effectiveness, inventory management, labor utilization, and customer service and delivery [32].

## 3.2 The Risks and Challenges of IoT

From the big picture of IoT applications, it is evident that the new technological opportunities have personal, organizational, cultural and social implications. The IoT is now used to increase household and workplace efficiency. The sensors can communicate and take action such as place an order for food when the fridge is empty, notify Smartphone when the washing machine is done, turn off few lights if there is no one around and so on. The possibilities are expanding day by day. However, the effects of malfunction of these devices can be costly too as we depend much on IoT. The malfunction can create inaccurate data, and can cause dangerous outcomes if that information is being used for the decision-making process in automated home or production. The door can be left unlocked, or the entire production can be stopped. All of these IoT devices are adding some values to make things easier for individuals as well as businesses; however, they also cause risks [33].

Tony Bradley [34] defines IoT as most security concerns of five (5) security threats for 2015-16 and these are: "

1. IoT: The Insecurity of Things.
2. Sophisticated DDoS Attacks.
3. Social Media attacks.
4. Mobile Malwares.
5. Third-party attacks."

Herath and Rao defined Risk as the "perceived severity of a threat" and the "perceived probability of the occurrence" of the threat [35]. The IoT devices collect a large amount of data of individual behavior and thus carry a high potential for privacy risks. The identification of a person and his behavioral patterns with these data are now rising concerns in the field of IoT security. As these devices are increasingly used in all areas of personal life and organizations. For example in health care sector, these devices can be used to collect and store private information of patients and share via a router to a communication device (Wi-Fi or cellular). This information could be used to identify disease correlations and support new treatment options as well as the remote monitoring of the treatment process. If the data is misleading or collected improperly or an illegal device captures data, then the privacy could be at risk [33].

Smith [36] discusses that the real cost of IoT is privacy. It is what the people cannot even imagine risking yet. Most people still believe that the Antivirus can protect them from the potential threats of searching web pages. Nowadays, a lot of free digital services such as Facebook, Twitter, LinkedIn, and Snapchat are offering smartphone applications and a lot of personal information, are feeding into these applications. People do not care about it but eventually; organizations make money from this personal information such as advertisements.

Interactive advertisement is also having a new exposure as IoT is becoming an essential part of our lives. People are enjoying the benefits of EZPass (faster and discounted road toll payment), supermarket apps (gives discount and free food) and many other applications. Although these applications might not be interested in our personal information, these applications can provide better service when they have some personal data. For example, if an application has the data that a user is vegetarian, then it will guide them to a vegan stores or restaurants. By providing a little information, the user might enjoy constructive suggestions about what to buy or which stores to go with special deals [37]. But most of the users are unaware of that fact that powerful corporation and government agencies track them as they interact with their environment. This way exposure of unauthorized information becomes possible as the current web-advertisement framework allows third-party cookies to track individual's web browsing history [38].

Zettaset Inc. [39] identifies that distributed computing allows data to be processed anywhere when a resource is available. That means it can create multiple copies of the data and process at different servers which add more complexity. Besides, Butun et. al. [40] argued that whenever web service models and unsecured communication channels are adopted, security risks grow significantly.

Butun et. al. [40] also argued that Anomaly detection has been widely adopted security measures, but IoT has many sources generating data and cloud services that are offered at geographically remote places. The sensor data may be noise, inaccuracy or distorted. That is why traditional anomaly detection techniques do not work well in IoT.

An employee can try to connect to the corporate network to access its resources or information anytime and unintentionally the device could spam into the cloud system. The IT Department of Financial Organizations is in a more vulnerable situation. These organizations had to securely keep all the transaction and related

**Table 3.1:** Threats and Attacks on Mobile Devices [42]

| Threats and Attacks | | Description |
|---|---|---|
| Sniffing | | Tapping or eavesdropping, e.g., GSM A5/1 cracked |
| Spam | | Email spam and MMS message spam, e.g., unsolicited MMS |
| Spoofing | | Spoof "{*Caller ID*}" *or MMS "{Sender ID}", e.g., spoofed MMS messages from 611* |
| Phishing | | Steal personal information using a spoofed target mobile application |
| Pharming | | Redirect web traffic to a malicious website and followed by more specific attacks |
| Vishing | | Voice phishing by utilizing VoIP technique |
| Data leakage | | Unauthorized transmission of data, e.g., mobile virus ZitMo |
| Vulnerabilities of Webkit engine | | Vulnerability allowing attackers to crash user applications and execute code, e.g., the Webkit vulnerability revealed by CrowdStrike |
| DoS | Jamming | Jamming radio channel |
| | Flooding | MMS message flooding attacks and incoming phone call flooding attacks |
| | Exhausting | Battery exhaustion attack |
| | Blocking | Use smartphone blocking functions to disable smartphone |

communications, including transaction documents, voice and written messaging [41]. Moreover, many of the tablets and Smartphones designed for consumer use lacked the capabilities to meet rigid compliance requirements. Any information, when downloaded into the mobile, the organization does not have any clue how and why it is being accessed. That leads to a severe security threat to all the organizations.

As the Internet of Things (IoT) is becoming common aspects of our lives by connecting technologies together, this uncontrolled growth rate of connected smart devices represents yet another opportunity for exploitation in the online world. In fact, enterprises cannot ignore the potential impact that this ecosystem will take over information protection and privacy [18].

Recently, Wang et. al. [43] argued that introduction of cloud with smart devices integrates many functions such as emails, notes, various documents, calendars which also includes the sensitive information referred to organizations or staff office. The devices also may carry critical business information. This information may also include sensitive personal data:

- Personal data such as home address, phone number, pictures, sin number, and contact lists, and so on.
- Correspondence, business information such as emails, text messages, MMS messages, call logs.
- Credit card info, secret credentials such as usernames and passwords.
- Business planning files on flash memory or memory card.
- Corporate documents such as word documents and spreadsheets.

- Geographical location to detect user current location.

Unfortunately, the information is not limited to this list only. The central data storage is always attractive to the hackers. These data could be an easy target leaving no trace behind. The cloud is still vulnerable to the attacks. Wang *et. al.* [42] summarized a list of threats and attacks in Table 3.1. These threats and attacks are usually carried out by malware. These malwares disguise itself while downloaded as usual mobile applications such as games or simple applications and installed in the device. Virus, Trojan and Spyware are the three top categories of mobile malware [44]. For the mobile platform, Trojan and Spyware are the dominant malwares. The US Department of Homeland Security (DHS) warned law enforcement, security and government workers against using outdated versions of Google Android, claiming that 79% of all mobile malware targets the platform [45].

Romers [46] assessed the risks of the interaction of cloud computing and mobile devices in his research and explained that organizations have to consider everything from data contamination to user habits to the activities of criminal syndicates. The issues the author raised in his research field:

- Consumer devices are not designed to adapt with rigorous data security. These devices either lack advanced security features or security features are disabled by default.

- Employee's personal photos and files are stored on the device. However, that increases the chance to mix up personal and official documents. The careless configuration of backups or file copies, personal files could end up in corporate file servers and vice versa.

- Malware's are acting like mobile applications and attacking mobile devices. IBM predicts malware will grow by 15% per annum for next few years.

- Employees now check their emails or work during evenings and weekends. While they manage that, they mainly use their personal smart devices by bypassing firewall inspection, so attackers are now targeting phishing attacks during non-business hours.

- Peoples are losing devices in every 3.5 seconds in the US. Those devices could have the credential to access organizations, resources and could contribute to a severe privacy issue.

- Sharing bunch of files and private cloud applications are practiced by peoples nowadays. These inspections and repairs are popular and inexpensive but do not provide enough security to be trusted with corporate information.

Weinberg et. al. [31] stated that more data would be generated with the development and implementation of IoT environment. Now the approximate data amount is four zettabytes (i.e., $10^{21}$ bytes) whereas it will be 40 zettabytes in 2020 [47]. But the question is who will own the data? These data are stored in a free storage and are being used by the third-party or providing trending results to the third-party organizations.

Providers, organizations, and manufacturers need to make some standard policies as these data are personal data.

In conclusion, the significant challenges in IoT include lack of awareness of people, unwillingness to maintain proper policy, not having advanced security policies, lack of protection, unknown threats and hence along.

## 3.3 IoT Protocols

In order to support the constrained application layer protocols, three (3) IoT standard protocols have been proposed. The protocols are Data Distribution Service (DDS), Message Queue Telemetry Transport (MQTT), and Constrained Application Protocol (CoAP). An overview of these protocols are discussed below:

- *Data Distribution Service (DDS)*: Object Management Group (OMG) [48] develops Data Distribution Service based on publish-subscribe protocol for real-time M2M communications. This architecture suits well to the IoT and M2M communications and supports 23 Quality of Service policies including security, urgency, priority, durability, and reliability.

  DDS architecture consists of two layers: Data-Centric Publish-Subscribe (DCPS) and Data-Local Reconstruction Layer (DLRL). DCPS is responsible for delivering the information to the subscriber whereas DLRL, an optional layer, acts as the interface to the DCPS functionalities.

  Figure 3.1 shows the conceptual model of DDS. Publisher spreads the data, and DataWriter is used by the application to interact with the publisher to define the data and its type of the data. The subscriber receives data from the publisher and delivers them to the application. Each subscriber has a DataReader to access the received data, and a topic is identified by the data type and a name. Topics relate DataWriters to DataReaders, and data transmission is only allowed within DDS domain [49].

- *Message Queue Telemetry Transport (MQTT)*: MQTT is developed by IBM to connect embedded devices and networks with applications and middlewares. The connection operation uses a peer-to-peer routing mechanism (one-to-one, one-to-many, many-to-many) which enables MQTT protocol as an optimal connection for the IoT and M2M.

  MQTT also uses a publisher-subscriber pattern to provide both transition flexibility and simplicity. MQTT is suitable for devices with low memory and bandwidth. It is built on TCP protocol, so it is fast and reliable. Figure 3.2 shows the architecture of MQTT. A subscriber would be interested in a device that registers for specific topics. The topics are published by the publisher and broker are responsible for notifying the subscriber if the topic is in its interest area. The MQTT protocol represents an ideal messaging protocol for the IoT and M2M communications and is able to provide routing for small, cheap, low power and low memory devices in vulnerable and low bandwidth networks.

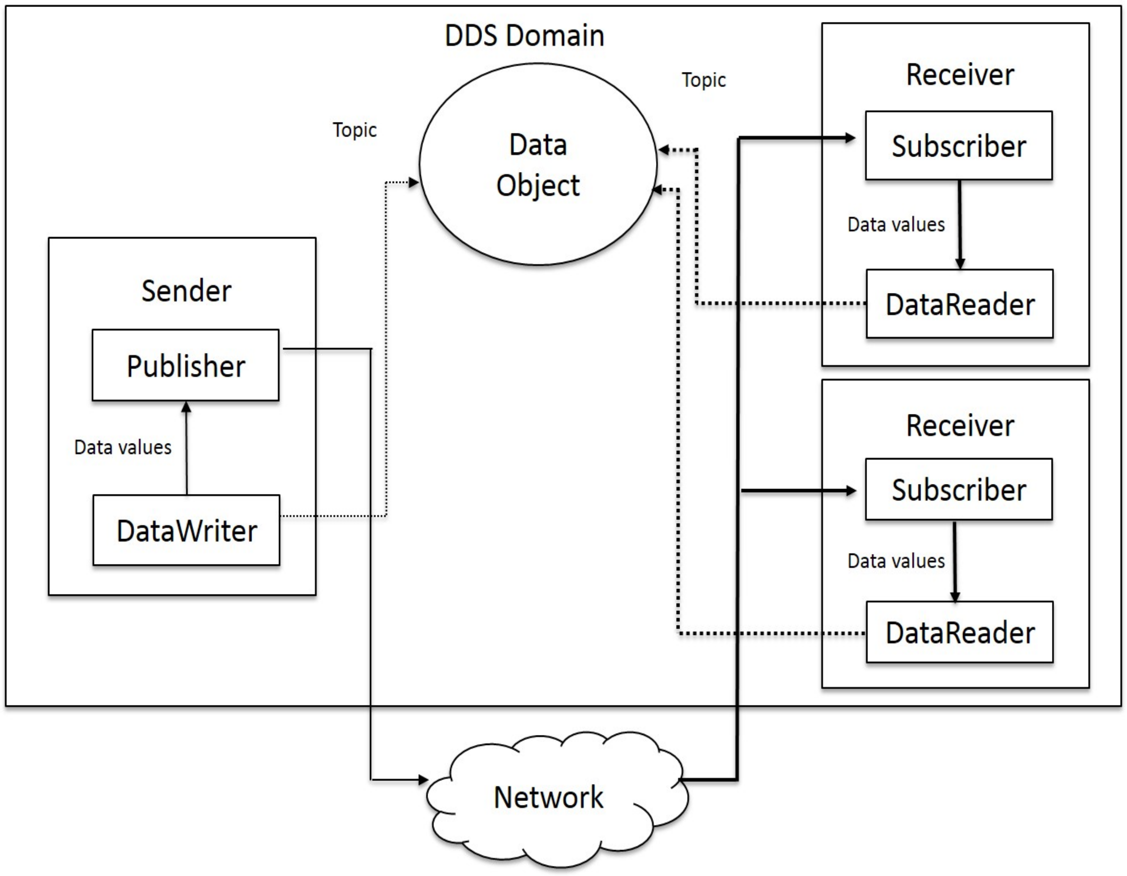


**Figure 3.1:** The Conceptual Layer of DDS [49]

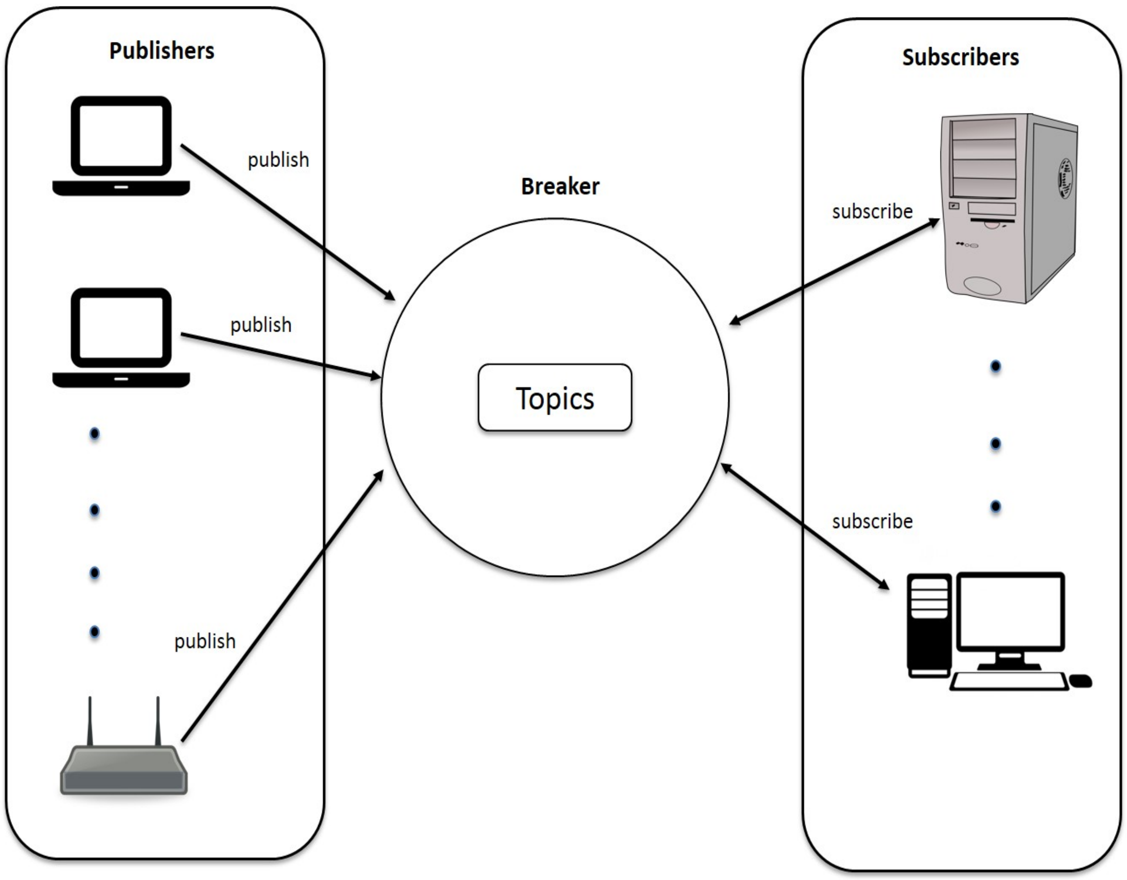
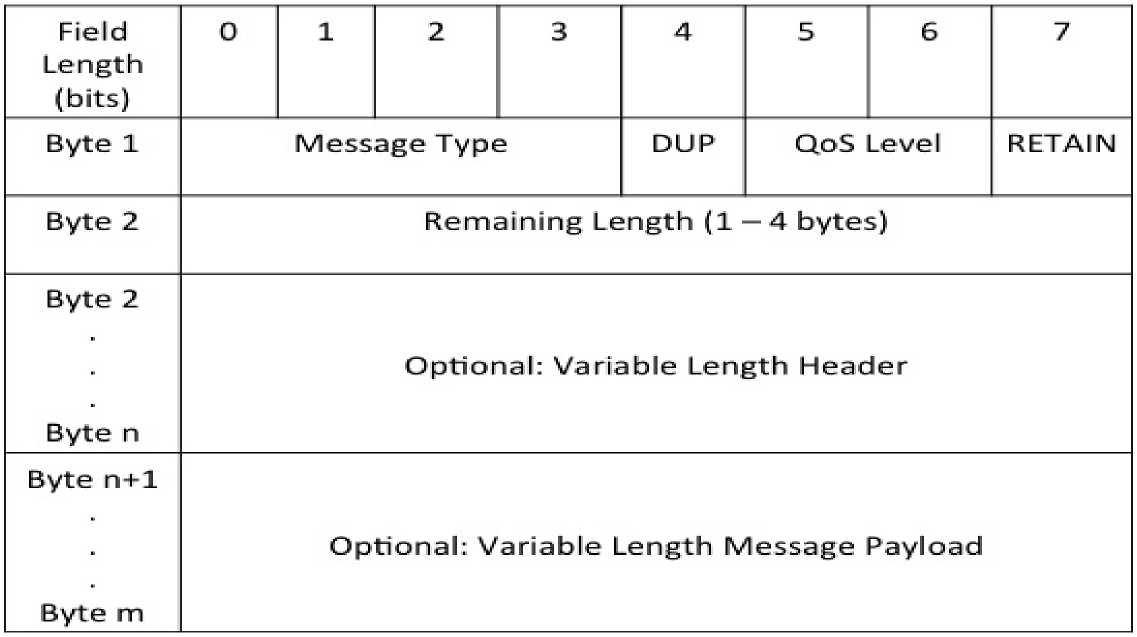


**Figure 3.2:** The Conceptual Layer of MQTT [49]



| Field Length (bits) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Byte 1 | Message Type | | | | DUP | QoS Level | | RETAIN |
| Byte 2 | Remaining Length (1 – 4 bytes) | | | | | | | |
| Byte 2 . . . Byte n | Optional: Variable Length Header | | | | | | | |
| Byte n+1 . . . Byte m | Optional: Variable Length Message Payload | | | | | | | |

**Figure 3.3:** The Message format of MQTT [49]

Numerous applications such as health care, monitoring, energy meter, and Facebook notifications are using MQTT.

Figure 3.3 shows the message format of MQTT protocol. The first two bytes are fixed header. The first byte contains message type, DUP flag, QoS level and RETAIN. CONNECT, CONNACK, PUBLISH, SUBSCRIBE are the main message types, and DUP indicates if this message is a duplicate. QoS flag indicates the level of assurance of a PUBLISH message. The Retain message informs the server to retain the last received PUBLISH message and submit it to the new subscriber as the first message. Byte 2 contains the remaining length field which shows the remaining length of the message [49].

- *Constrained Application Protocol (CoAP)*: The Constrained Application Protocol is an application-layer specialized web transfer protocol designed to provide support with constrained nodes and constrained networks in the Internet of Things. CoAP is designed in such a way that it would interface easily with constrained nodes and provides support to integrate HTTP web requests. This integration considers that the nodes are low powered, and the interaction model is similar to the client/server model. A request made by CoAP client is sent to the server using HTTP verbs (GET, POST, PUT, and DELETE). The server then sends a response with a Response code including/excluding a resource representation.

  CoAP follows the statelessness constraint as its architecture is on REST framework. However, CoAP uses datagram-oriented transport such as UDP to deal with these interactions asynchronously. Although UDP is unreliable, this is done logically using a layer of messages that support optional reliability. As CoAP requests and responses are carried out in separate messages, to ensure reliability every message has to have one message type. Each request has an additional bit to describe the message type. The

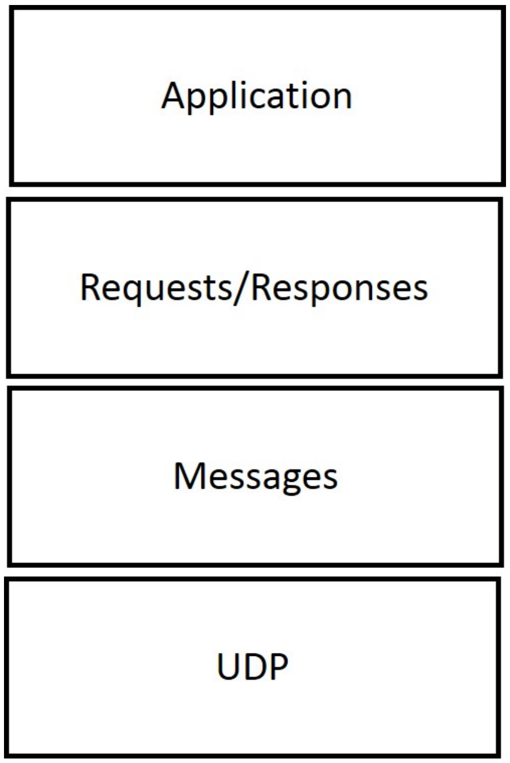


**Figure 3.4:** The Abstract Layer of CoAP [50]

message types are:

– *Confirmable message (CON)*: A confirmable message requires an acknowledgment. The client will keep resending a CON message until it receives an ACK message with same message ID.

– *Nonconfirmable message (NON)*: A nonconfirmable message does not require an acknowledgment such as GET request.

– *Acknowledgement message (ACK)*: An acknowledgment message confirms that the earlier message was received and processed.

– *Reset message (RST)*: Similar to the acknowledgment message but it also confirms that the recipient could not process it due to network loss or other reasons.

The interaction between CoAP client and server are carried out in CoAP messages which include either a Method Code (CON, NON) or Response Code (ACK, RST). A request is carried out to another device or server with CON or NON message type and depending on the availability of the recipient, and the response will be sent to the sender in the resulting ACK message. If the sender sends a CON message and does not receive ACK from the receiver, then it is going to resend the same message until it receives an acknowledgment message. That is the reason, if the receiver cannot send the desired response, it will send an empty acknowledgment message so that the sender can stop resending the request. Afterward, when the desired response is ready, the recipient sends a new CON message to the

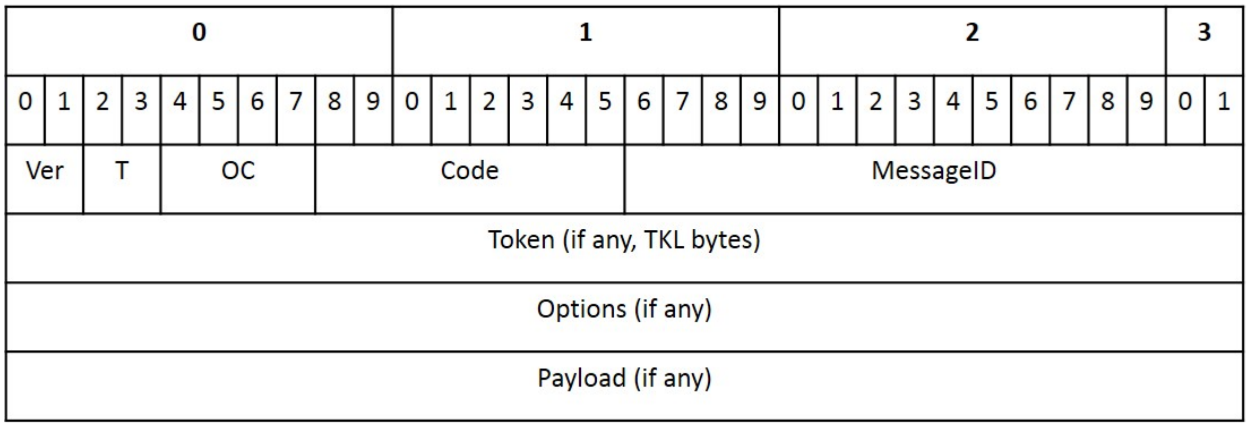


**Figure 3.5:** The Message Format of CoAP [50]

sender and then the sender has to send back an ACK message to the recipients. This process is known as the separate response. If the request is sent with NON message type, then the response is sent using a new NON message.

From figure 3.4, CoAP can be logically defined as a two-layer approach. One of the CoAP messaging layers deals with UDP and the asynchronous nature of the interaction whereas the other one deals with request/response interactions using the method and response code. As CoAP is based on the exchange of compact messages, the messages are encoded in a simple binary format. Figure 3.5 represents the binary structure of CoAP message format. The CoAP packet format has a maximum length of 1400 bytes whereas the header has a length of 32 bits (4 bytes). First 2 bits for version control, next 2 bits for message type and last 4 bits for token length. Shelby et. al. [50] provides the details of the messaging format of CoAP.

### 3.3.1 Why CoAP

The traditional request/response model is not suitable where clients are interested in having a quick response to the representation of a resource over a period of time of time. In these scenarios, MQTT works well as it uses peer-to-peer communications. However, CoAP offers real-time solutions along with maintaining the properties of REST to push resource representation from servers to the interested client. CoAP clients can be initiated as Observers and register their interests using the GET request with a special 'observe' option activated for one or multiple resources. CoAP is also popular because of the integration of 6LowPAN, easy portability with HTTP, and UDP to support low connection overhead. Thangavel et. al. [51] compares the performance of MQTT and CoAP in terms of end-to-end transmission delay and bandwidth usage. The experiment shows that MQTT performs well when the data packet loss is low. However, CoAP outperforms MQTT when

- Messages are small in size and data lose rate is under 25% [51].

- In Smartphone environment, CoAP's bandwidth usage and round trip time are smaller than MQTT [52].

CoAP is different than HTTP protocol, but it maintains REST architecture. HTTP protocol is not compatible with small sensor devices. CoAP follows the statelessness constraint as the requests and responses are separate. CoAP's request and response semantics are similar but not identical to HTTP's. It is a web protocol that fulfills M2M requirements in constrained environments. CoAP also has some other features which make it special to build any system for IoT. These are:

- Low header overhead and parsing complexity.
- Asynchronous message exchanges.
- URI and content-type support.
- simple proxy and caching capabilities.

A CoAP device can send multicast UDP message to its surroundings to observe who else is around. It can start exploring and communicate with other devices following REST architecture and without any human interaction. All of the things can be considered as the world of APIs. Home automation, controlling and monitoring everything with the mobile device is possible with CoAP [15, 50, 53, 54].

## 3.4 Access Control System

Access control allows authorized users to access resources. After authentication, access control identifies the user and controls access of that user. The access control depends on the role of that user and sharing a key between multiple devices. At first, role-based access control models have been discussed. Some of the role-based solutions also integrate key exchange mechanism. The descriptions of each type of role-based access control are list below:

- Attribute-based access control (ABAC): access is granted based on attributes of the users or devices;
- Discretionary access control (DAC): object owners decide access policies;
- Mandatory access control (MAC): access policies are determined by the system; and
- Role-based access control (RBAC): users are assigned different roles, and each role has its operational permissions.

### 3.4.1 IoT Solutions in Access Control System

In the IoT environment, access control is important to make sure that only trusted users can update device software, access sensor data or command the sensors to perform an operation. Access control resolves data ownership issues and enables new services such as Sensors As a Service, where sensors provided data to customers. Access control enables to share IoT device data with authorized users to allow both predictive maintenance and protection of the sensitive data [55]. That is why access control in IoT context is becoming more and more important security element and access control in IoT context is discussed below.

1. Usage Control(UCON): Park and Sandhu [56] developed the idea of encompasses traditional access control that focused on authenticated users, trust management that covers authorization for unknown users in the open network (Internet), and digital rights management to protect client-side resources. UCON enables control over both centrally controllable environment and environments where central authority is not available.

   Zhang and Gong [57] argues that UCON is effective in IoT environment because it introduces mutability. Traditional access control systems do not support run-time modification. The permission is made before the access control, and it can not be changed during access control. The decision is made by comparing the security levels between the subject(user) and the object. But, UCON provides the facility to change the values of the attributes of the subjects and the objects on runtime which supports the flexibility and security needed for IoT environment. The changes will be effected in the next transaction.

   Zhang and Gong [57] also propose an access control architecture that applies usage control model for the IoT. The UCON integrates authorization, obligations, conditions, continuity, and mutability to provide support to control usage of resources in IoT environment. Based on fuzzy theory, this approach proposes access control policies and process based on the abstraction of UCON model and assessment model. However, there is only a few experiments present that does not provide enough evidence or confidence of using this approach in IoT nodes.

2. Capability Based Access Control(CapBAC): In CapBAC, the access control provides user an unforgettable access token to access a resource. As long as a user's access rights are updated, the resources rights do not need to be updated. This is a benefit in the IoT context. In a decentralized environment, the majority of the published IoT systems include a server that grants access token. A user presents this access token while they are connecting directly to the IoT devices [56].

   Gusmeroli et. al. [58] provides a detail description of the Capability Based Access Control (CapBAC) system for managing access control within the European FP7 IoT@Work project.The CapBAC model works as follows: a device (Device 1) receives the token with its capability written. Then it (Device 1) presents this token to another device (Device 2) that it wants to access. The device (Device 2) provides a service presented to the token through verifying the token. This CapBAC system has some features such as validity period, assigned rights, delegation depth and flexibility. However, this access control model does not support on mobile devices yet.

3. Distributed Capability Based Access Control(DCapBAC): Ramos et. al. [7] analyzes that the centralize access control has a separate PDP (Policy Decision Point) server that helps to update the security

policies any time. But it is hard to consider the context of IoT and ensure end-to-end security as PDP makes the decision. To overcome this problem, they proposed an access control method considering the context of the IoT device. This access control solution supports the management of certificates, authentication, and authorization process. The access token is provided with ECC digital signature in JSON format. That token contains information about the resource to access and action that can be executed. Although this model may require additional control logic to transmit context values and also to suffer from transmission delays, end-to-end access control validation has been achieved.

Zhang et. al. [59] propose a solution for distributed access control system to ensure privacy preserving. The device owner provides the client with only one token, and sensor nodes will validate this token before reply to any of the query. Their approach does not allow reuse of tokens by multiple clients. However, this access token verification is based on verification of digital signature using RSA algorithm that is time and resource consuming for low powered devices.

4. Role-Based Access Control(RBAC): Discretionary Access Control (DAC) and Mandatory Access Control (MAC) are traditional security models that apply to different applications and are usually based on user-group. So these models are not suitable for open network [60]. In RBAC, users are associated with roles and these roles are associated with a set of permissions. Each resource is also associated with permissions and RBAC decides whether user's permission is matching with resource's permission. Adding access rights to users is easy as long as existing roles are used [56].

   Zhang and Tian [60] proposes an extended role-based access control model for the IoT using the context information. The main goal is to enhance the security for web services' applications and produce a perfect mechanism to control accessing of data in IoT environment. It uses the conjunction, disjunction, and negation operations to express context constraints.

5. Attribute-Based Access Control(ABAC): Attribute-based access control provides a different approach to authorization and access is based on a user having specific attributes. This approach provides a better access control system by combining not only user attributes but also other data (IP Address, Mac Address, Location). Rather than using the role of an authorized user, ABAC combines multiple attributes to make a context-aware decision for a sensor at run-time.

   Adaptive access control [61] is designed by ISSA End-to-End Trust Working Group and is based on ABAC (Attribute Based Access Control). It takes access decisions based on attributes of the request. Access control policies are defined to trust the request and minimize the risks. The access policy varies depending on the trust of the requesting user and device. However, it is not always possible to compile the trust policies with all of the nodes instantly. It is based on ABAC but can be extended to capability access control and role-based access control.

   YE et. al. [2] aims to establish a mutual authentication by using an efficient Elliptic Curve Cryptography (ECC)-based authentication and the attribute-based access control policy. As this process ensures a secure communication between user and nodes and uses low storage and communication overhead, the resource-constrained problem of IoT application layer can be solved.

Various key-based access control solutions also have been proposed for different IoT application scenarios. The following researches have been done to improve key-based access control protocol in IoT.

1. Zhao [62] presents smart business security IOT Application Protocol intelligent Service Security Application Protocol (ISSAP). It reduces the overhead of data resources and uses a data packet encapsulation mechanism which combines cross-platform communications with encryption, signature and authentication algorithm to establish a secure communication system in IoT.

2. Kothmayr [63] designs a standard and RSA-based security architecture with two-way authentication for the IoT. It is based on existing Internet standards, and Datagram Transport Layer Security (DTLS) protocol, which is placed between transport and application layer. This authentication is performed during DTLS handshake and exchange of 2048-bit RSA keys. The extensive evaluation shows that this architecture provides message integrity, confidentiality, and authenticity with enough affordable energy, end-to-end latency, and memory overhead.

3. Roman [64] classifies key management system into four categories: key pool framework, mathematical framework, negotiation framework, and public key framework. Most of the key management system do not comply with IoT. The reasons are:

   - Key pool framework suffers from insufficient connectivity. Most key chains are constructed by randomly extracting a subset of keys from a key pool. The size of the key chain is usually much smaller than the size of the key pool. So there is always a chance that two different key chains will not share a common key, and that is not acceptable, as the server needs to accept any connections from the clients it knows.

   - Mathematical key pool protocols make use of the deployment knowledge to optimize the construction of their data structures. Deployment regions can be partitioned into small areas. Key chains of the small areas can be constructed in a deterministic way rather than selecting a random set from the key pool. However, this methodology cannot be used in IoT context as client and servers are usually located in different physical locations.

   - Negotiation key pool uses the wireless channel, and it has a feature to negotiate effectively for a common key. As some protocols increase the power of transceivers to send information to its nearest neighbors or limited range of transceiver to negotiate in the same cluster, client and server nodes might belong to different networks in IoT, and that is why it cannot be used in IoT context.

   - Combinatorics key pool cannot be used as it suffers from connectivity and scalability authentication issues.

However, The Blom [65] and the polynomial schema [66] are few of the other which can incorporate key pool management protocols. The computational overhead is quite small for these IoT scenarios in comparison to a Public Key Cryptography (PKC) operation [14].

4. Both Pranata et. al. [67] and Ning [68] presents frameworks using Public Key Infrastructure (PKI) to authenticate, authorize and access control in IoT environment. The objectives were to overcome the lacking of traditional Internet network security and ensure minimal computing resources.

5. Zhen-Qiang et. al. [69] proposes a transmission model with signature-encryption schemes. A security architecture along with security protocols in the ONS (Object Naming Service) has been designed to address the IoT security requirements. A local ONS server can have a temporary certificate from trusted authentication server (TAS) after Root-ONS validates its capability from TAS. Then with that certificate, Local-ONS are able to inquire as many times as it wants within the time frame. In this protocol, the Remote Information Server uses multiple encryption layers with routing node's public key and encrypted data is decrypted in each node until it reaches to the local-ONS. Its hop-by-hop encryption/decryption behavior is fragile in terms of security [14].

6. Lee et. al. [70] enhanced the existing RFID system security and proposed a lightweight encryption based authentication protocol for IoT environment. Instead of using encryption method such as hash function, they have chosen encryption method based on XOR manipulation for anti-counterfeiting and privacy protection.

7. Wang et. al. [71] proposes an ECC-based access control scheme for the wireless network based on public key cryptography. Yeh et. al. [72] also proposes a secure authentication protocol based on the intractability of ECC logarithm problem.

Although there has been a lot of researches in the area of the access control system in IoT environment, these specifically target the problem of lightweight cyphering in pervasive environments and improving standard protocols. The key-based access control is secured and popular because it aligns well with low-level sensor devices. These sensor devices share keys to communicate and recognize each other.

However, all of these devices are connected to create more services, and these services produce a huge amount of data. The data can not be monitored or stored using these low-level sensors, and that is why it has become essential to incorporate cloud-based platform into IoT environment. Cloud has the capacity to compute, store, provide services, and software that could be easily provision as Infrastructure-As-A-Service and Software-As-A-Service. Applications that are accessing IoT sensors might need to store massive data, process data in real time, and high-speed network to stream audio or video and using cloud services these requirements could be fulfilled [73]. That is why a centralized cloud-based system is proposed in this thesis.

From the recent studies, it has been observed that different access control systems have different capabilities. MAC and DAC based access control are not suitable for open networks, and the benefits of key-based access control system are constrained to the decentralized system. There have been new access control systems proposed such as CapBAC, DCapBAC but these still under development [7, 58]. In Role-based access control system, the roles are developed with permissions and users are assigned with one or multiple permissions. But in IoT context, each resource might have different access rule, and it is expected that by 2020, there will be 50 billion devices [74]. Rather than creating a single permission set for each resource, the best way is to define a set of policy or configuration file for each resource. That is why attribute-based access control system is considered in this thesis as an access control model. The benefits are significant:

- Access rules are easy to implement and lightweight [56].
- The policies can be changed on run-time [56].
- While accessing a sensor device, the cloud does not need to look into all roles. It can only concentrate on the specific configuration file.
- Access decisions are managed centrally [75].
- Implementation is simplified by the incorporation of "policy enforcement point" [75].

There are already some similar architectures available. ISSA End-to-End Trust Working Group proposed an Adaptive access control in February 2012 [61]. Adaptive access control enables the power of ABAC and overcomes the fixed mechanism of traditional access control system. This framework is composed with Policy administration point(PAP), policy enforcement point(PEP), policy decision point(PDP), policy information point(PIP) and other features. The request from the device is coming with two kinds of attributes and the service forwards this request to the PDP for further processing. Now PDP processes Front end attributes using policies defined in the policy repository and also sends the 3rd party attributes to the PIP. Now PIP processes 3rd party attributes and sends back to the PDP. Then PDP grants access if the criteria are met and PEP let the device to access resources. In adaptive access framework, the request is coming from the device directly, and the evaluation started later. As PEP is responsible for initiating the evaluation procedure and the services are low-level devices, the security could be breached easily by using any powerful devices. Also, there are two places (PDP, PIP) where the policies are evaluated which increases the chance of high latency. However, in this thesis, the request is first received by the cloud and cloud is solely responsible for evaluating the request. The request reaches to the services after the cloud permits. Also, the low-level devices are only responsible for communicating with the gateway server and implements simple CoAP messaging protocol.

YE et. al. [2] uses an attribute-based access control system to enable flexible, and fine-grained access control. They have used an ECC-based mutual authentication system as it requires less memory to occupy and strong ability against the attack. This mutual authentication system has three phases: Initialization, Mutual Authentication, and Key Establishment Phase. In the initialization phase, the Base station generates the private key, public key, hash function, elliptic curve and its parameters. In the next stage, the user selects a base point and generates a public and private key for authentication and the node verifies these keys and generate a secret key for that particular user. Later on, the user again generates a secret key and shares that key with the node. If the hash value of the user secret key is matched with the secret node key, then the user gains authorization. After authentication, the attribute-based access control system is implemented to restrict the resource access. Mutual authentication is still requiring four steps to authorize which increases the chance of high latency and the device only gets partial access to the network. The mobile device is connecting to the resource node directly which also has a risk factor. However, in this thesis, the authentication is done by third party and authorization is only focused. The device needs to authorize from the cloud at first and then the cloud access the sensor nodes. This procedure ensures security because the gateway server can only send connection request to the cloud. After establishing a secure connection, the gateway sends the data packets.

The proposed system overcomes the limitations of existing systems and can provide more secure system architecture to establish an IoT network. The proposed system can be implemented in different applications of IoT systems. Home automation solutions and Healthcare solutions are discussed in below.

### 3.4.2 Trust Models in IoT

Building and gaining trust in embedded sensors has become one of the interesting research topics. A trust value can be established by the interaction with other entities, context values, recommendations and other factors. A well-built trust system allows internet users to share information and sensitive data without worrying about security issues.

IoT is largely dependent on WSN and ATRM [76], an agent-based trust and reputation management scheme for WSNs. In ATRM, the mobile agents are designed to travel and run over the entire network, and the trust and reputation management carried out locally with minimal overhead in terms of additional messages and time delay. Another agent-based trust model is presented in [77] a watchdog scheme. This scheme is used to observe and analyze the behavior of nodes and broadcast their trust ratings among sensor nodes. The agent nodes are responsible for monitoring, computing, and broadcasting the trust ratings. Most of the recent studies are based on similar trust relationships and routing decisions [78, 79]. Chen et al. [80] proposed a trust management model based on fuzzy reputation for IoT environments. The tasks were done in three steps. At first, the identity of nodes is evaluated against trustworthy authority. The second step includes data processing and routing. Lastly, evaluation combines component forms the new trust degree from old trust value and the indirect information of the third party node. However, these trust models are not effective when there is a high sensor activity because the trust has to generate for each level whenever a new node is joining or removing from the topology.

Sicari et. al [14] argues that the trust concept is a complex notion and has different meanings in different contexts. A variety of definition exists, and the biggest problem with many approaches is that they do not grant on the certain method to the establishment of metrics and evaluation methodology. The satisfaction of trust requirements is related to the identity management and access control issues.

Several works [81, 82] focus on trust level assessment of IoT environment and smart objects. These smart objects have heterogeneous features and build social relationships such as friendship, ownership, and community. The authors assume that most of these objects are exposed to public areas and public networks, hence vulnerable to malicious attacks. The malicious nodes aim to break the core functionality of IoT by self-promoting, bad-mouthing and good-mouthing.

Bao and Chen [81] proposed a distributed, encounter-based, and activity-based trust management protocol. Two nodes can rate each other whenever they are interacting and also can provide a recommendation about other nodes. Honesty, cooperativeness, and community-interest are used as reference parameters. A similar approach is followed  [83] where the authors integrated social networking concepts into IoT. The objects are capable of establishing social relationships in an autonomous way and a build a reputation-based trust mechanism for the SIoT, which can deal with particular types of malicious behaviors. Lacuesta [84] also proposed a secure distributed ad hoc network based on social network context. Each node has an identity in the peer-to-peer interactions and modifies the trust of other nodes from their behavior to create a trust chain among users.

The fuzzy approach is also considered in several research works [85, 86] to build and evaluate trust models. Mahalle et. al [85] considered that the traditional access control models are not suitable for the decentralized and dynamic IoT scenarios. They proposed a Fuzzy approach to the Trust Based Access Control (FTBAC). The trust scores are calculated from the FTBAC factors such as experience, knowledge, and recommendation. FTBAC framework includes device layer (includes all devices), request layer (responsible for collecting FTBAC factors), and access control layer (decision-making process) and the simulation results guarantee flexibility, scalability, and energy-efficiency. Wang et. al [86] presented trust evaluation based on three layers: sensor layer (physical devices), the core layer (access network and the Internet) and the application layer (includes p2p, cloud, etc.). The authors used fuzzy set theory and formal semantics-based language to perform trust evaluation.

Other proposals [87, 88, 89, 90] exist to build and evaluate trust model in IoT environments. In this thesis, the trust model is straightforward (linear matching), and the proposed architecture is designed in a way that any suitable trust model can be integrated later.

### 3.4.3 IoT Applications

There are other various challenges when it comes to implementing solutions for the risks of IoT. The primary requirements of building and running an IoT environment are having an access code of conduct, establishing security programs and requiring management rules [91]. A solution may not be suitable for every person, industry or even for every employee in the company. Smartphones could be distracting because of the notifications from all the IoT devices. Smartphones are now more powerful and could control all of the low-powered connected IoT devices. So the private data could be in danger if somehow an unauthorized Smartphone connects to the network.

Researchers are working on to supply a more beneficial answer to mitigating threats for IoT. Different IoT access control solutions along with guidelines have been discussed in this section.

### Home Appliances Solutions

Sivaraman et. al. [92] proposed that device-level protections need to be augmented with network-level security solutions to minimize the privacy and security issues of automated smart home appliances. Network level security can be implemented across the entire range of IoT devices where device level security is for a particular device. The security implementation and practice is highly variable depending on the device capabilities, and manufacturers. They have proposed the use of software-defined networking (SDN) using context values (time-of-day or occupancy of the house) to implement dynamic security rules. They have focused on network level security whereas this thesis focuses more on system architecture level.

Godha et. al. [8] also discusses that there are not many ways to identify home automation devices. Although there are existing standard security protocols yet, this field still has room for improvement. Scalability and low memory are the significant problems to store keys which make private key cryptographic solutions inefficient. They have proposed an access control system which does not focus on authentication. It mainly focuses on how much access a device should get. Whenever a new device is given access to the internet networks should have a particular tag by the centralized router. That tag defines the access level of the device. The device that has the highest level of privileges can perform READ/WRITE and other operations on that device or their data. However, it does not include dynamic access so that any device will have similar access level all of the time. In this thesis, the access is defined and evaluated in the cloud and includes dynamic access control system.

### Health-Care Solutions

Savola et. al. [93] discusses that security and privacy for e-health in IoT environment is a challenge. The number of chronic-disease patients is increasing worldwide, and the cost is relatively high. The most effective treatment is self-care which includes IoT devices. Sensor devices and well-managed data collection are central to self-care which makes security and privacy requirements high in health care system. The main contributions of their work are analyzing security risks of an e-health self-care system that contains medical IoT sensors, communication and storage solutions, processing and presentation of the data, and the appropriate interfaces in between. They have also proposed a heuristic to measure the risk impacts of threats on the security controls. The proposed architecture can also be integrated with e-health services too. If monitoring devices are set up for a patient and these devices are only capable of access by the authorized personal then the proposed architecture is suitable for this scenario. Being on the same network will not help the attacker to gain control of the monitoring machines as the access control only allows the attributes from the authorized machines.

**Table 3.2:** Suggested Safeguards against the threats [94]

| | |
|---|---|
| 1. | Use encryption for database backups. |
| 2. | Encrypt the on-line production database information. |
| 3. | Avoid vulnerable OSs on server. |
| 4. | Grant access privileges that are specific to job requirements. |
| 5. | Maintain a record of employees who work longer than the allocated shift time. |
| 6. | Make a provision of taking digital signature or other biometric signature of employee as per the importance of transaction. |
| 7. | Providing guidance to staff against the possible risk of information leakage. |
| 8. | By applying data leak prevention techniques. |
| 9. | Use of data-centric security approaches instead of secure perimeter approach. |
| 10. | Details of career and psychological profiles of persons hired for information security purposes. |

**Guidelines and Safeguards**

A list of safeguards against the threats is defined by Kumar and Singh in Table 3.2. They proposed a proactive security management system that integrates patches, assets, and configuration management systems. The treats could be worms, virus, criminal insiders, threats from internal sources, and a discontent employee or partner and so on. They keep the MAC address of every registered device with an employee and create a tuple of the form ($Ei$, $Dij$, $MACij$). $Ei$ means an employee, $Dij$ means the employee $Ei$ can save multiple devices from $1, 2, 3, ...j$ and that is why each employee may have multiple Mac addresses $MACij$. Each Mac addresses $MACij$ belongs to a device $Dij$. Then create a provision to store every login entry for every transaction made and manually checks if it falls into any of the case studies they mentioned. They have also done an empirical study on a company and collected data for seven days to validate their research [94].

Smith [36] addresses that the balancing act is needed to balance the vast volume of data. These data generated by IoT devices are like a double-edged sword of value and risk. People need to balance the benefit of using IoT and protect their privacy in a hyper-connected world. The FIPP declares several guidelines to protect privacy as well as enjoy the benefits of IoT. The guidelines are:

- **Collect limited and specific data**: There should be a limit to the collection of personal data, and the purpose of collecting data need to be specified. IoT devices should obtain data or data collection from IoT devices should be legal.
- **Data Quality**: Personal data should be relevant, accurate, complete and up-to-date.
- **Security Safeguards**: Personal data should not be disclosed, made available or otherwise used for

any purpose. Reasonable safeguards should protect data.

- **Openness and Individual Participation**: The data should be used in constructive ways, and any individual has the right to obtain personal data from a controller.

- **Accountability**: A data controller should be accountable for protecting users data.

There are a few more solutions exist that include both enterprise and mobile device components. Titze et. al. [95] proposed a Security Service Architecture (SSA) that copies user's smartphone on the enterprise side and checks for the security. Chung et. al. [96] produced a novel architecture called 2-Tier Access Control (2TAC), which uses double layer access control (one in the device and another one is in the cloud) along with device security profiles, anti-virus/malware scanners, and social networking. Ekberg et. al. [97] also proposed a secure environment named as Trusted Execution Environments (TEEs). It separates the implementation of an application into secure and insecure parts by using a protected area in the central processing unit. TEE provides space isolation. But it does not offer policy enforcement. However, data are stacked away in the mobile devices. TrustDroid, an Android application developed by Zhao and Osono [98], analyzes other applications to prevent illegal access to corporate information.

Establishing a secure IoT environment is still under the investigation. The proposed architecture in this thesis includes dynamic attribute-based access control, cloud, REST framework for HTTPS interaction, CoAP for device level interaction, and two layer trust calculation to create a bi-directional secure environment for IoT.

## 3.5 Smartphones

Smartphones are one of the most recent trends in the new era of IT. Smartphones are characterized by having two essential features; being multi-modal, spatially aware, along with having a development platform [99]. Multi-modality serves the key purpose of keeping the device continuously connected through a combination of different connections. These capabilities allow the device to switch between 4G, Wi-Fi, and Bluetooth depends on the context and situation.

Smartphones are also becoming more powerful compared to the previous cell phones. The latest Smartphones are using a high tech processor such as A8 for iPhone 6+, Snapdragon, 805 for Samsung Note 4 with at least 3GB ram [100]. Due to that high powered computing facility, the field of mobile computing can focus more on issues about the implication of software running on these nomadic computers. Users are more probable to be performing tasks and also doing other things. Whenever any data is downloaded into that mobile, the mobile itself can process this information in the background. This data could send to other web address without slowing down the Smartphone. Thus, Smartphones have the ability to simplify our work as well as perform operations that could bring threats to our data, including privacy.

In IoT, Smartphones can play a vital role. Smartphones has the computing power and memory which most of the IoT connected devices do not have. With Smartphones, it is possible to control cars, home automation [8], Smart TVs, etc. There are several apps also to monitor and control these smart sensors. Now, these devices have less computing power and memory to validate or store records of accessing data, and that is why compromised Smartphones or gaining illegal authority with a Smartphone can put sensitive information and privacy at risk [101, 102].

### 3.5.1 Mobile Device Security Issues

As it is already discussed that mobile device technology is changing rapidly, business mobility capabilities are also becoming more popular and critical. "Business Mobility" is defined by Nokia as giving employees the desired "freedom" to collaborate and transact business outside traditional workplace and work-time. Employees spend more time interacting and communicating with customers or clients and less time at their desk. They do not have to wait to go to their workplace to do the work. Also, instant communications now enabling the option to response instantly to any requests and real-time customer supports [103].

Over the year, many solutions for mobile device security has been proposed and used. Some of the approaches are described below:

- The addition of trust hardware for distributed mobile devices. The Root Trust Model is proposed to improve the trust between mobile users and mobile devices to authenticated booting, platform integrity, and data access through a set of hardware and software mechanism [104].
- IT security policies in the mobile devices can be editable. It can be loaded into when the device booted up. The organization can build a structure of the policies and force the user to use it.

Marsh [105] also explains a mechanism to make a valid judgment by three components such as user, location, and task. Marsh describes users' perspective by explaining the following trust level phases. The phases are:

- **Imprinting:** an initial state where devices build a robust model of trust and behavior using users' identifiers.
- **Nurturing:** The trust between user and devices are enforced in this phase.
- **Growth:** The trust grows as the user use the device properly.
- **Repair:** The relationship between user and device needs to restore if something goes wrong.
- **Use:** While accessing sensitive information, device's comfort level is also an important factor along with user's credentials.

From a *location* perspective, the location affects device's comfort depending upon whether it is in the comfortable zone or discomfort zone. From *tasks perspective*, a routine task increases device's comfort level and any tasks that are never seen before or executed before is considered as a discomfort task.

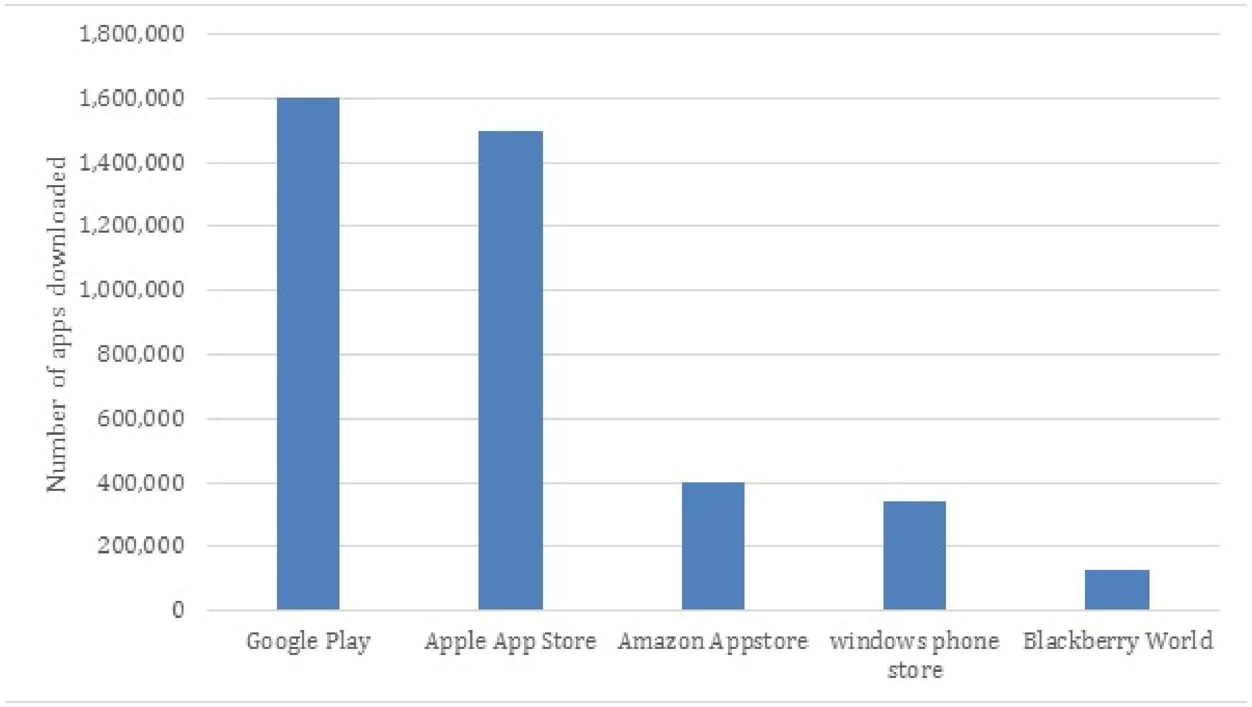


**Figure 3.6:** Number of apps in July 2015 [107, 108]

Harris and Patten [106] also discussed that security specialists of Information systems are familiar and knowledgeable with a desktop computer and firewall systems. However, the primary concern is now with the increasing popularity of bringing Smartphones into the workplace. Instead of using Windows operating system and related applications, these devices use different OS and applications that are made and control by the operating system's parent such as Apple and Google. Security experts are well known of traditional security threats, but the introduction of IoT raises new concern about safety and security of enterprise's business, customer and employee data and information. The primary concern regarding device security is when user or employee downloads information into the device or connects to the cloud service from a compromised device.

In 2015, the number of mobile applications available in app stores is shown in figure 3.6. It is estimated that 179,628 million application will be downloaded in 2015 and will reach to 268,692 million in 2017 [108]. However, the rise of mobile apps comes with the increase of malware. Wessel et. al describes malware as eavesdropping and modification of remote communication, installation of malicious applications on the device, and exploiting (known) vulnerabilities to gain access to higher privilege levels, typically root access [109]. 99% of malware detected in 2012 was written for Android [110]. Malware for Android increased by 350 percent in 2012 [111].

The transition from personal to professional use of mobile devices is driving more stringent security requirements. F-Secure Lab's [112] *2014 H2 Mobile Threat Report* pointed out that malware is the primary threat facing mobile devices. Malicious applications can be found in the app store, and employees can install these applications if not careful enough. These malware exploit vulnerabilities in other target applications or even in the operating system too. It may steal sensitive data, corrupt the integrity of data and transaction and overall usability of a device while executing. This execution is done in hidden mode. Recent studies [113] have shown that it is not particularly difficult for a programmer or hacker to implement a distribute malware using app store or using other distribution channels. Although few countermeasures at the market-side mitigate adverse effects, it does not eliminate malicious applications completely [114, 115, 116, 117]. Cisco 2014 [118] reports that 98% of malware written to target text messaging. These type of malware can send text messages in the background without notifying the user. 78% of malware detections fall into this category and cost the average user $9.99 a month [119].

Another major issue affecting all platforms are fake applications [120]. It looks and behaves like a "real application". The user does not have any idea that it contains malicious code that might use to steal or track user's activities. Arxan Technologies, a security firm to provide application protection and anti-tamper solutions for a variety of defense and commercial software, reports that top 100 of Android applications and 92 of the top 100 iOS applications have fake malware versions on third-party application sites. These third-party distribution channels are easy to access for Android users to download and install the free version of paid applications. For Android, users do not need to jailbreak their phones but for iOS, the device has to be jailbroken. However, there are not many third-party distributions for iOS, so it is quite difficult for iOS users to find such applications but not impossible [121].

Olga Gadyatskaya, Fabio Massacci, and Yury Zhauniarovich introduces two emerging operating systems, The Firefox OS and Tizen Mobile Platforms. The native integration of web applications is the distinguish features of these OSs compared to the mainstream mobile OSs. As these two systems are Linux based, the authors discussed their security designs and features in details and compared the differences between these systems and Android as well [122].

## 3.6 Context Information

### 3.6.1 Context Information Definition

Context information is defined as any information that can be used to characterize the situation of an entity [123]. A situation can be a time, a location, a heading direction or a social context. Schilit [124] divided *context* into three categories. These are:

- Computing Context: It can be current network type, bandwidth, mobile devices memory and storage.
- User context: It can be current location, a user profile, etc.
- Physical Context: It can be current temperature, wind direction, etc.

A good number of built-in sensors such as Cameras, GPS receivers, acceleration sensors, light sensors, ambient sensors, orientation, proximity sensors, level sensors and much more are available in Smartphones these days. This Smartphones hardware meets the requirement for implementing a context-aware application to support the trend towards a highly mobile workforce. Mark Weiser [125] was the first person who conveys the idea of context-aware computing. Abowd et. al. [126] categorized context-aware applications into three types. These are:

- Presentation of information and services to a user.
- Automatic execution of a service.
- Tagging of context to information for later retrieval.

The recent growth of context-enabled applications, researchers now proposing context provisioning systems that help to build context-aware applications more efficient. Context Toolkit [127] is a context GUI widget, a context provisioning architecture that provides a platform to discover and provision context information to different entities [128], and middleware that allows for providing context information between consumers, and providers [129] are some of the examples of context provisioning systems.

### 3.6.2 Security Solution Based on Context Information

Sheng et. al. [130] compared context information with real life scenario. Context information has been used to identify whether somebody tells a lie. For example, when someone wants to affirm nationality of a refugee, the most commonly used method is to ask that person to speak or write claimed native language. Context information can also be used to identify the inappropriate activity of an authorized user.

TaintDroid [131], an extension of the Android mobile-phone platform that tracks the flow of privacy sensitive data through third-party applications, assumes that download and installed from third-party applications are not trust-able and monitors during execution and when accessing private and personal data. It is important because mobile phone operating systems currently provide general control of accessing private information but provide a little insight into how data is being used. For example, if a user permits an application to access GPS data, the user has no way of knowing if the application will send the location data to a location information- based service, to advertisers or any third party entity. Thus, the user needs to trust blindly that the application will handle their sensor data properly.

Context information is also using to implement Cyber Security Defense. Sheng et. al. [130] proposed a method to build a protection system for cyber security defense using context information. Initially, they have simulated different faulty scenarios to know fault data. Afterward, based on that knowledge, possible fake data is generated that can cause malfunction. These data is utilized as training sample to set up a probabilistic neural network (PNN). The protection system detects faults in the communication network according to the measurements of a single or couple of instrument transformers. All of these calculated measurements of the substation will be collected and feed to the constructed PNN to detect vicious fault induced with bad data.

Oliver [132] conducted a study on 17,300 BlackBerry devices and observed an average interaction time per day of 101 minutes. 80% of the usage sessions take 90 seconds or less. This high percentage of short communication is explained as *checking habits: limited but repetitive access of dynamic content providing some form of informational "reward"* such as: checking email or Facebook [133]. Verkasalo [134] examined 324 Smartphones users' contextual usage patterns. They found that the device usage was diverse in office and home context.

Soikkeli et. al. [135] studied the relation between mobile device usage and end user context using 140 Smartphones. While not distinguishing locked and unlocked usage, they found that usage sessions are longer in home context while more frequent in an office context.

Context information could play a vital role to implement security system in IoT environment where the cloud has minimum knowledge of its clients, and one client with powerful processing capability can gain control of other low powered devices.

## 3.7 Cloud

Cloud Computing, the long-held dream of computing as a utility [136], refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. It is massively scalable to provide easy and quick support around the world. Secondly, it can be delivered as an abstract entity to the clients, while at the same time offering them many different services. Ultimately, it is driven by the economics of scale. Services are provided on demand and revolve around a clear for profit business model. It is not surprising that the largest developers of cloud computing technology are Microsoft, Amazon, and Google. For example Azure from Microsoft and Google App Engine [137] from Google are some of their products offered today.

### 3.7.1 Security in the Cloud

The recent research work shows that cloud computing security issues can be divided into two categories: cloud storage security and cloud computing security [138].

Cloud environment has several risks associated with it such as (Injection Attacks, Cross Site Scripting, Broken Authentication and Session Management, Failure to Restrict URL Access, etc.). Access to the resources of the cloud is often provided through a website interface. Thus, attacks to the interface include cross-site scripting. Risks that are more generally applicable to cloud connection include broken authentication and session management [139]. Smartphones can be dangerous due to broken authentication. We always have a tendency to keep ourselves always logged in.

A survey done by Ahuja and Komathukattil [140] presented some common threats and associated risks to the clouds. They have also proposed that *Intrusion Detection System (IDSs)* can be applied to detect attack patterns and heighten security measures by looking into and monitoring user actions, web traffic, transaction logs and access logs.

Xiao and Xiao [141] presented an organized review of security issues in the clouds. They have taken the survey based on attribute-driven methodology. Confidentiality, integrity, availability, accountability and privacy-preservability are the primary attributes. Threats are categorized into each attribute, and each category is reviewed along with the corresponding defense mechanisms.

Aguiar et. al. [142] focused on the cloud computing security. The study includes several issues such as authentication and authorization, virtualization, web services, accountability and availability regarding server storage and data computation security. They have also discussed the possible techniques and the mechanism for achieving proper accounting, storage privacy and public verifiability of outsourced data and computation.

Pearson [143] also focused on privacy, security and trust properties of cloud computing. They have discussed the current security state of cloud systems. They have also considered the countermeasures such as Responsible Management and Privacy Protection to build the trust between cloud and users.

Zhou et. al. [144] provided a survey on the security and privacy issues of many cloud computing providers. Security is studied with focuses on availability, confidentiality, integrity, control and auditing characteristics and privacy by listing out-of-date privacy acts.

Vaquero et. al. [145] provided profound insight into IaaS clouds security and Rodero-Merino et. al. [146] have also presented a survey of the security state in PaaS cloud environments. Armbrust et. al. [136] have proposed Anything-as-a-Service (XaaS) by joining IaaS and PaaS. By XaaS, they have referred to the fact that cloud systems can support and offer anything and everything in the form of services.

### 3.7.2 Security between Mobile and Cloud

The adaptation of the IoT in the enterprise and personal life is not only to an increase in productivity but also to increase in vulnerability and security threats. The use of Smartphones is growing to access cloud backend or data storage. A report, Cyber Risk Report, done by HP in 2012 states that mobile platforms represent a major growth area for vulnerabilities [147].

Grispos et. al. [148] demonstrated a vulnerability in iCloud syncing mobile applications, such as cloud, Dropbox and so on. Hackers can extract logs and retrieve deleted files from these applications because they act as a mirror for what is in the cloud. As these applications can contain proxy view, an attacker can access the files without accessing the cloud directly. Info security [149] shows the fact that data leakage can be possible even after deletion of the files/applications or a factory reset.

Jailbroken smartphones are common nowadays, and it enhances the security threats because malware can now reach into the kernel parts more quickly. Jailbroken smartphones allow users to install less secure applications by accessing other parts of the operating system that, in general, is not accessible. By jailbreaking the phone, users can download applications from underground distribution channels. So it becomes easier for a malicious application to reach sensitive components of the operating system, including previously protected decrypted data [150].

Li and Clark [150] also discussed an attack based on Short Message Service (SMS). They have discussed two types of attacks. The first one has the objective of sending premium-rate SMS to offshore accounts or mass SMS spam advertisements with the intent for phishing. The second one has the intention of using the Smartphones to be part of a highly efficient and stealthy botnet.

There is a possibility that Phone recycling could also leak not only private data but also company-owned data. From YouGov/Blackbelt survey results, it is found that approximately 59% of mobile phone recyclers are not concerned about private data left on their recycled mobiles [151]. Therefore, organizations cannot overstate the continued increase of having company-owned data in mobile devices or accessing data from mobile devices [147].

In this section, the state-of-the-art on cloud security issues is discussed briefly.

## 3.8 Web Services

Web services are now becoming the dominant paradigm for e-business. Due to the advanced mobile internet (4G, LTE) and open WIFI connection, e-business and other cloud applications now provide web services that could establish a connection between mobile devices and cloud.

Web services mainly expose their functionality via Extensible Markup Language (XML) or JavaScript Object Notation (JSON) and APIs. The HTTP request fails to guarantee data integrity and security. Most SaaS vendors deliver their web services APIs without transaction support that makes it complicated to manage the data integrity across multiple SaaS applications. On the other hand, WSDL is a language standard for describing the functionality of a Web service, such as how it can be called, what parameters are expected for the input and the return values. However, WSDL permits invoking other operations and thus metadata spoofing attacks are becoming common. Metadata spoofing attacks are easily detectable if sound methods are used.

Service-orientated architecture (SOA) and web services are one of the most significant enabling technologies for cloud computing in the sense that resources (e.g., software, infrastructures, and platforms) are exposed to the clouds as service [152]. According to a recent study in Europe [153], the web approximately contains 30 billion web pages. Another 10 million new pages are being added every day. But only $12,000$ web services exist on the Internet. Among these web services, most of the web services are deployed with dependency problems [154, 155, 156]. According to Sheng et. al. [157], web services can be divided into two categories: operational behavior and control behavior. The operational behavior is application dependent and illustrates the business logic that supports the functionality of a web service. The control behavior is application independent and acts as a controller over the operational behavior and guides its execution progress. Recently, verification of web services has become an active and an interesting research topic.

The next generation of web service is heading towards highly collaborative application and introduces questions about traditional security mechanism, such as access control and cryptographic protocols. Traditional security mechanism fails to address new evolving security challenges (uncertainty, openness, and fraudulence). Based on that, Liu et. al. [158] propose a novel evaluation model of a web service by improving the SOA framework by incorporating a trust management module. Then they propose the trust evaluation model based on a compelling subjective logic by viewing trust relationships of service entities.

### 3.8.1 REST Framework

REST, a popular and modern web service mechanism introduced by Roy Thomas in 2000. It is lightweight, has a practical implementation of web services compared to WSDL/SOAP-based implementation. REST is an alternative solution for Smartphones or in a web browser, for which it would be too cumbersome to integrate heavyweight web services. When REST framework is applied to a web service, it brings beneficial properties, such as performance, scalability, and modifiability.

Mobile Devices use REST framework to access Cloud services. Instead of a device having to formulate complex XML tree structures, which are expensive in terms of memory and processing power. Simple HTTP verbs (HEAD, POST, PUT, GET, and DELETE) can be used as an alternative method. The response is also memory friendly and the standard and minimal HTTP error codes found in the header.

The REST architectural style describes six constraints. The constraints are:

- Client-Server: Separates clients from servers. The client does not know about the storage and server is not concerned about the GUI.
- Uniform Interface: Defines the interface between clients and servers. It simplifies and decouples the architecture and enable each party to evolve independently.
- Stateless: Every request from the server holds within it as all the information needed to service the request. Statelessness allows greater scalability since the server does not have to maintain, update or communicate through a session.
- Cacheable: Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later.
- Layered System: Refers to different layers when it comes to the back-end servers and architecture. Servers can be put in places that are in-between clients and other servers, to improve scalability, and enforce security policies more easily.
- Code-On-Demand: REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts.

These constraints, applied to the architecture, define the basis of RESTful-style [159].

Most importantly the stateless interaction, uniform interface, identification of resources through URI and using self-descriptive message are the popular characteristics of REST, and that is why CoAP also uses REST framework to interact between the server and the clients.

## 3.9 Summary

There are some existing approaches from the mobile device's perspective. Those are:

- Adding trust hardware for distributed mobile devices and trusted components for mobile OS.
- IT policies for mobile devices are editable. Security policies can be loaded into devices when the OS's starts up.

But those approaches are ineffective due to multiple devices held by a single user. Each device cannot comply with policies. And these policies are also static and cannot be modified later on dynamically.

In this literature review, researchers have shown that although IoT has a high potential to facilitate our life but still is a threat, and in coming days' organizations' and personal life will be under threat. They have also discussed how to provide the solutions, but most of the results are based on Key-based encryption. Equally, it is already noted that users might not feel safe to secure their mobile with MDM or MAM because they do not require to be monitored; whereas the proposed approach does not monitor or control the users' other activity and collects sensor data like any other apps such as Facebook, Twitter, etc.

This literature review also covered cloud security, mobile device security issues, web services secured interaction. Those solutions are working for a specific part such as web services can handle the security of data transfer, but that does not validate the trust level of the web services. If the application itself is tampered or compromised, cloud and web services do not have any way to see that. But the proposed system has to establish a trust module that can verify the authenticity of the web requests and can tell the cloud that if the request is made by the user so cloud can send the response back to the device. Also, these solutions do not consider IoT as a test case.

In an open environment where there is no central administration, building a secure model is the key to let low powered devices rely on the system. A successful access control system will be constructed on an initial trust value, updating the trust value based on interactions, context, and recommendations, and risk considerations.

# Chapter 4

# Proposed Architectural Design

In the previous chapter 2, it has already been stated that the question trying to be answered in this thesis is "How to trust a device while accessing IoT nodes in a connected network". A dynamic and bi-directional trust communication between mobile users, cloud, and sensor resources would be a beneficial method to satisfy the research goal. A dynamic attribute-based web authorization system is proposed to establish a trustful communication that can filter out bad requests by itself in IoT environment. That means it receives the web authorization requests from the mobile users to access sensor data and validate the users' request by evaluating context information in the cloud to ensure security and privacy of the sensor data. WebSocket protocol provides a secure connection from gateway server to the cloud application. The context information from users' devices helps the cloud to identify if the current user is the real user, which is the prime focus of this thesis.

In this process, the malicious requests are filtered out which means IT departments or System Administrators do not have to monitor the requests made from mobile devices as well as granting access to each mobile device. Cloud can avoid malicious requests without notifying the System Administrator. These requests will be kept in the spam requests to review later by System Administrator. In this proposed solution, the mobile interactions do not have to be monitored all times, and the interactions with the sensor devices will be safe.

The overview, architecture, data flow, system functionalities, and data model of this proposed system are described in the following sections.

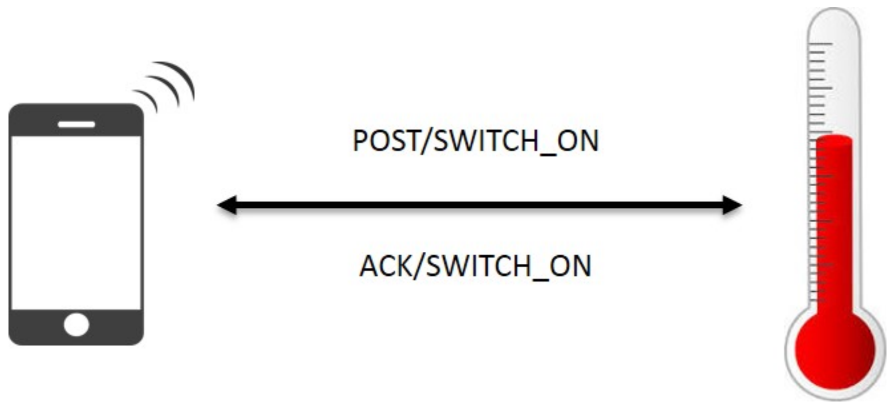


**Figure 4.1:** The Interaction between Mobile and IoT node.

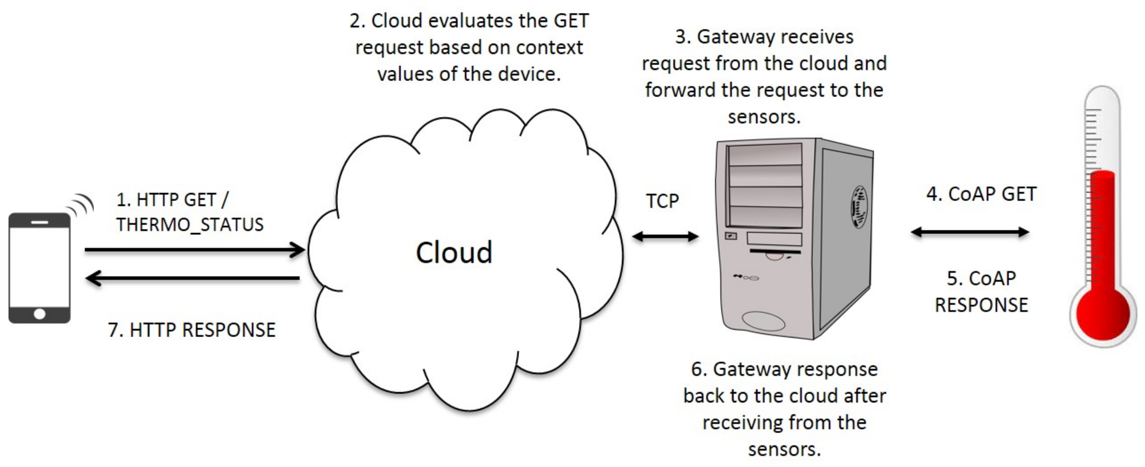

2. Cloud evaluates the GET request based on context values of the device.
3. Gateway receives request from the cloud and forward the request to the sensors.
1. HTTP GET / THERMO_STATUS
Cloud
TCP
4. CoAP GET
5. CoAP RESPONSE
7. HTTP RESPONSE
6. Gateway response back to the cloud after receiving from the sensors.

**Figure 4.2:** The proposed solution of the scenario.

## 4.1 Overview

Let us consider a scenario where a person has smart TV, Smartwatch, microwave oven, car, and many other devices connected to the Internet as described in [8]. The owner can control these devices with the Smartphone or get notification from those devices. Camera in a refrigerator can inform us what is missing, dryer can send notification after finishing drying, car can inform if it senses anything in the engine, safe box can send authentication code to our Smartphone to verify, notification from the security camera if someone is in a restricted room in the house, set the temperature of the house as shown in figure 4.1, and many more. Home automation has gradually led into a virtual dimension of augmented reality. But if somehow someone else gets into the system, the security of the home and personal belongings might be at risk. That is why a proper authorization system is recommended in home automation scenario.

In this thesis, a dynamic attribute-based access control system is proposed to strengthen the interaction in the IoT environment considering the above scenario. Figure 6.1 shows abstract view of the proposed solution. Each web authorization request is evaluated with corresponding policies and context information. The mobile device collects the context information using its sensors and calculates a trust value locally based on this contextual information. If the local trust value can gain minimum threshold trust defined by the administrator, then the web request transfers the query parameters along with the necessary context information to the cloud for further investigation. The cloud calculates the final trust value with the defined company policies. The cloud accepts the request based on the calculated final trust value. If the request is approved by gaining minimum threshold value, then cloud authorizes the mobile device to interact with the sensors for a particular time period. The interaction will happen between the mobile device and gateway server of the IoT nodes. If the final trust value does not meet the requirements, the cloud may ask for a secret question to verify the user. Figure 4.3 illustrates a high-level outline of the proposed architecture.

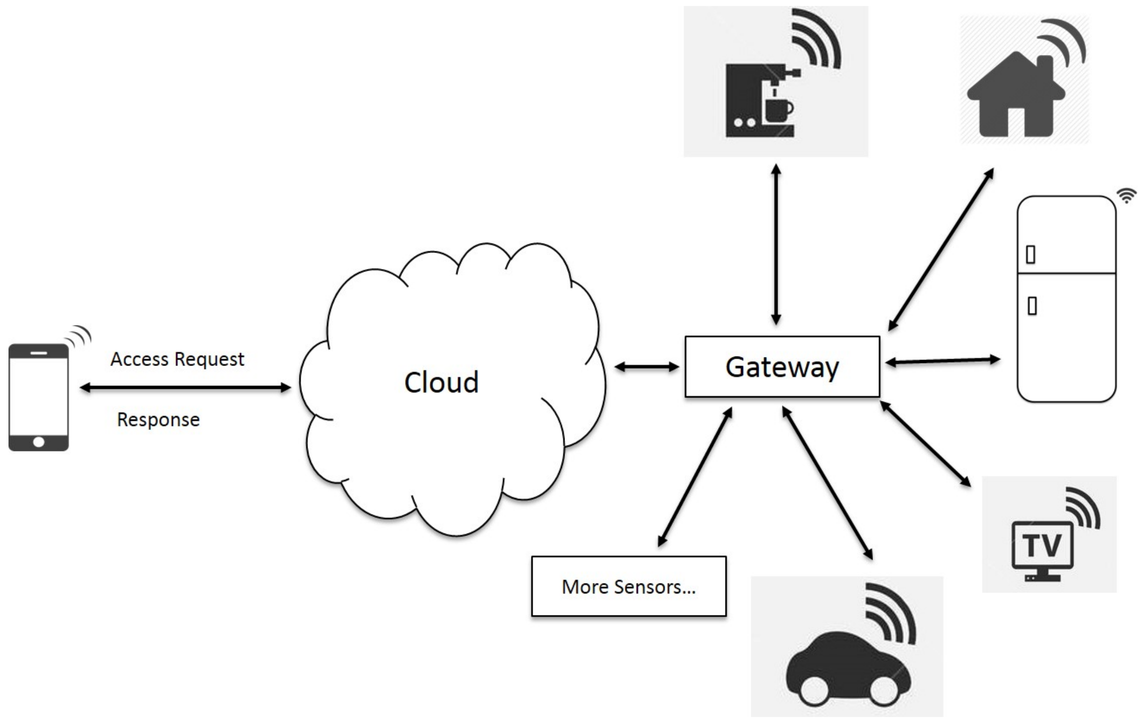


**Figure 4.3:** High Level Outline of Our Proposed System.

### 4.1.1 Overall Workflow

In this thesis, the goal is to control access by calculating an initial trust value in mobile/portable devices and then calculate the final trust value in the cloud. This two layer validation system is proposed to provide more powerful authorization between the mobile/portable devices and IoT resources through the cloud.

In figure 4.3, the mobile client sends a request to the cloud system for authorization purpose. The device collects location data, internet connectivity type, mac address and few other information as contextual information and calculates an initial trust value in the local device using these context values. The calculation of initial trust value is required to confirm that the device is not using any predetermined or static data to hack the system and the required data in the cloud is being sent from the mobile device. This will reduce the number of unnecessary requests and minimize load on the cloud. After calculating the local trust value, if it does not meet the minimum threshold value, then the mobile application shows an error message to the user and stops the request from being processed further. If the local trust value gains sufficient trust compared to the minimum threshold value, then the mobile client sends the request to the cloud for further processing.

Though the request is forwarded to the cloud server, mobile devices do not get access to the sensor information immediately. First, the cloud receives the request to validate these context information with defined policies and then decides if the mobile device can access those sensor devices. Here this cloud application is a separate application that holds the program/code to calculate final trust value and verify a request. For simplicity, proxy servers have been avoided in this thesis work.

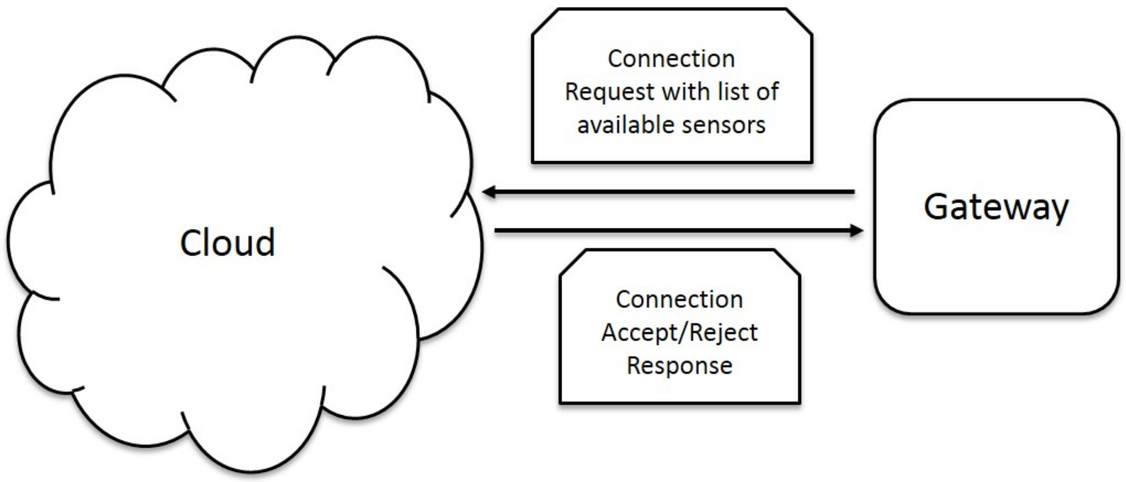

Connection Request with list of available sensors
Cloud
Gateway
Connection Accept/Reject Response

**Figure 4.4:** Gateway Server and Cloud connection.

The gateway server of IoT nodes is connected to the cloud as shown in figure 4.4. The gateway itself needs permission to join in the cloud. The cloud has the necessary credential to verify this connection and accepts if the gateway is valid. The connection between gateway and cloud is secured, and whenever cloud accepts a request, it sends the request back to the gateway. The gateway then creates a CoAP message and sends it to the destination IoT node. CoAP is a messaging platform in IoT environment. The IoT node sends back the response to the gateway, and the gateway sends the response back to the cloud as shown in figure 4.5. The reason to use the gateway server is to provide a secure channel between IoT nodes and cloud.

In this thesis, the proposed system is developed by extending the traditional authorization system in IoT environment. The traditional authentication system (username/email and password) is what we used to log into different social networking and web portals. In the beginning, the proposed approach verifies the request send by a user based on context information collected from the device. It calculates an initial trust value based on these context values (Network, Internet connectivity, and Location) in the device. If the measured trust value meets the minimum threshold value, mobile client sends the request to the cloud application to access these sensor devices. Now before allowing the request to access sensor data, the request made by the user is going to evaluate based on the calculated trust value, and policies defined by the company. The purpose of this evaluation is to compute a new trust value by considering context values, trust value, policies and allowing the device to access these sensors for a time period. The device can perform only the tasks that are authorized by the cloud. If a device has only READ permission, then it will not be able to WRITE or CONTROL those sensors. On the other hand, if the device has WRITE or CONTROL permission, a user will be able to control the sensor devices by their Smartphone. If the final trust value does not meet the minimum threshold value, then the cloud declines the request, and the user needs to depend on the further validation. For the further validation, the user might have to answer a secret question.

To describe the figure 4.3 more precisely, the proposed architectural model is classified into three tiers namely: the mobile client extracting sensor data and initial trust value calculation, the cloud application

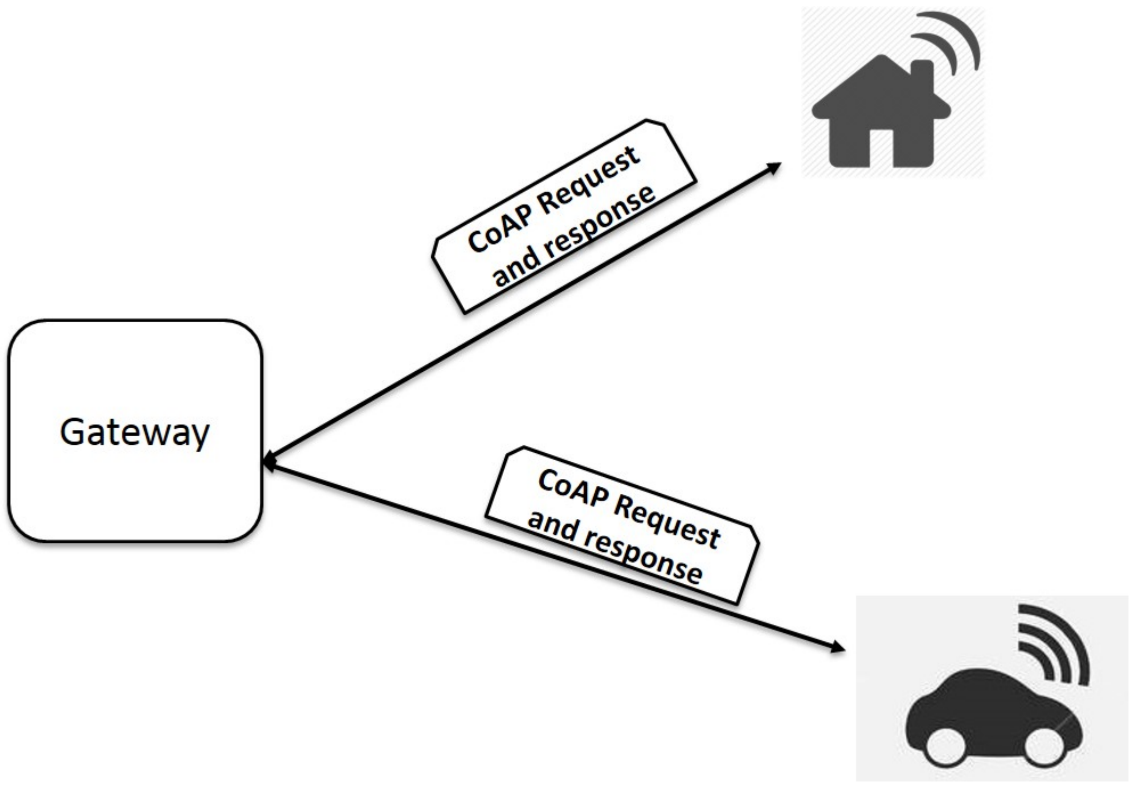


**Figure 4.5:** Gateway Server and Sensor Interactions.

to calculate the final trust value, and retrieving data from sensors. The contextual information is device's information that generates trust value, and the policy defined by System Administrator restricts the usage of application's resource. Before describing how these three components communicate, a discussion about using policies, and context values are shown below.

- Context Values: Due to various security risks, protecting information has become a challenging factor for many organizations. In this thesis, a plan is developed to integrate contextual information from the mobile clients to reduce risks. So that the proposed system can have an idea not only who and when but also how, where and why they are accessing the cloud applications.

  Context means surrounding conditions and the circumstances or events that form the environment within which something exists or takes place. Context has been used to identify or evaluate a situation approximately. Context information can simply answer the "W5H" question. The context information provides an answer to the cloud application that is interested to know during the authorization process.

  The importance of using context information to minimize IoT risks are significant. A detailed discussion has been covered in the literature review. The traditional authorization system is quite unacceptable in IoT environment. The traditional systems are more dependent on boundary or key based security control, and this boundary is not available now due to the introduction of IoT. IoT breaks the barrier and opens the IoT resources to control via Smartphone. That is why it is important to know how, why, when, where, who and what the request is trying to access, so a preventive solution can be designed.

  The access control system extracts contextual information from mobile devices. Mobile devices have multiple sensors that can collect the necessary contextual information periodically. The application receives this information from the sensors when needed and calculates the initial trust value using these context information. In this thesis, the following sensor information is considered:

  – Internet information: Which Internet provider the user is using? Usually, a user engages with one provider. Does this account receive a simultaneous request from a different internet provider?
  – Time: What is the time of accessing the resource?
  – Location: What is the current location of that device? The location is the most important aspect because the device and the authorized user have to be in the same location. Is it locating the user in a different location where they do not have any access?

  The internet, location, and time are the three items of information that help to design the system and later determine the common pattern of using the sensor resources of a user.

- Policy: Policies can be defined in various ways. Suppose if someone has access to all of the sensor devices and makes the request, but the system does not find enough evidence, then it might restrict the access. It may decline the request completely, or it may allow a limited access control depending on the company's policy.

  Nowadays, with rising IoT risks, having some static policies or traditional authorization system is not going to ensure privacy of the data. The policies need to be well defined and dynamic. The owner might have access to many sensitive data but based on current context; the access should be approved. If a user is trying to access the camera in the bedroom from a Smartphone or any secured information from an unusual location or time or network, then the system should restrict the access immediately. The user's device could be stolen or used by a different person and thus could lead to the violation of the integrity and privacy.

  That is why in the attribute-based access control, the users need to define their policies carefully and precisely. The more comprehensive policies the user is going to define, IoT resources are going to be more secured and well maintained.

  In this thesis, the focus is on authorization policies.

Figure 4.3 shows how the three components are linked together and have their tasks. Figure 4.6 shows how the mobile/portable devices perform actions. The steps are defined below:

- At first the mobile registers the sensors to collect context values. The background services notify the application whenever the sensor receives an update. By this way, the application keeps its sensor values up to date. In this way, whenever the user is making a request, rather than putting the application on hold to collect the sensor data, the application can start processing these context values at an instant.

- At the second step, the application gathers context values by using background services. The context values are stored in the mobile application. The application does not need to store the values in the database. The proposed system does not hold the previous data because the mobile application is not planning to use the previously used data to calculate the trust value at the initial level. It only depends on the current context values.

- At last, the mobile application calculates trust value based on the collected context values. Location, Internet Access, and Internet type are the main criteria to calculate the initial trust value. An HTTPS request is then sent to the cloud if the locally calculated trust value has met the minimum threshold value.

Figure 4.7 shows how the cloud application performs its actions. At first, the web request comes to the cloud application from a mobile client. The mobile client sends the package in JSON format as the proposed system follows the REST framework. Instead of accessing the sensors, the authorization takes part in the cloud. The cloud receives userID, password, necessary context information and the initial trust value calculated from the mobile device. Although the context values have been used to derive initial trust value, still the system requires few of the context values to match with the defined policies. Policies related to the context values are only considered for simplicity. The cloud application loads all the policies, and all of the policies are sorted out with the corresponding sensor key. Each sensor might be associated with a different level of policies. Then the cloud calculates the trust value and the user can access these sensor data according to the access control.

Now after the final trust value is calculated in the cloud, the cloud creates a session with a session key. Mobile client can access the sensor data until the session expires. After the expiration, the mobile client needs to authorize the device again. If the final trust value does not meet the threshold value, then the cloud will send a message for further validation. The User will be asked to answer predefined questions to verify himself. The threshold values and questions are customizable which make the architecture more dynamic.

Figure 4.8 shows the interaction between cloud, gateway, and the registered sensors. After accepting the

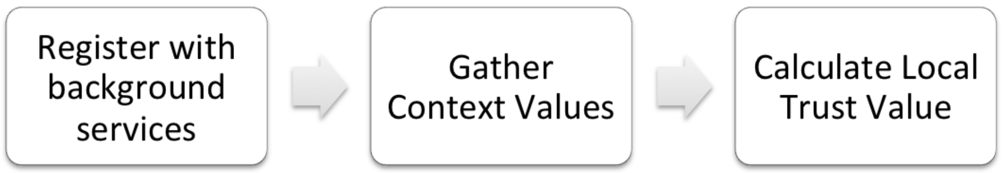

Register with background services
Gather Context Values
Calculate Local Trust Value

**Figure 4.6:** mobile Component

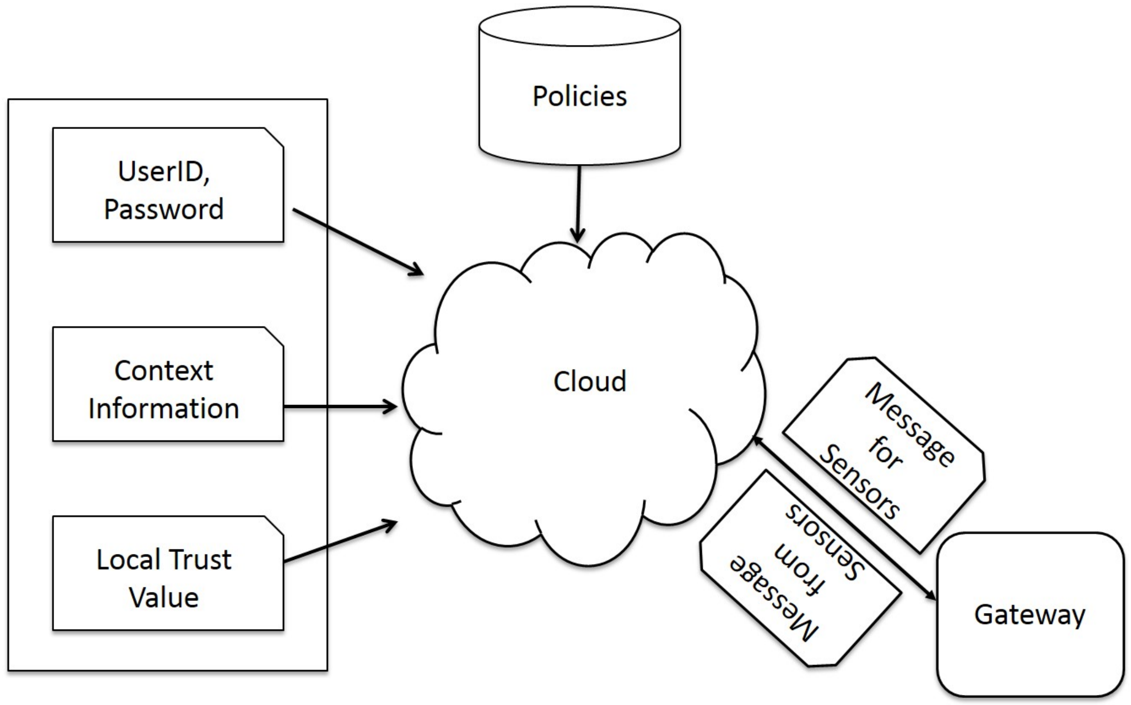


**Figure 4.7:** Interaction of Mobile and Cloud Application

web request, Cloud sends the request to the connected gateway which contains that IoT node. Now gateway accepts this request from the cloud and converts it into a CoAP message and transfers this message to the IoT node. The gateway has all information of all of the registered sensors, and this gateway is solely responsible for message transferring. The IoT node sends the response to the gateway and gateway sends the response back to the cloud.

In the proposed approach, the cloud application is applied to act as a dynamic shield to protect sensor devices from being misused. The objective is to neutralize the risks arises due to IoT and overcome the problems of traditional authorization and firewall system in IoT environment.

## 4.2 Architecture

The proposed architecture is explained through both physical and logical perspectives.

### 4.2.1 Physical Architecture

The proposed system is divided into four components from the physical architecture perspective: the mobile clients, the cloud, the gateway server, and IoT nodes.

- **Mobile Clients**: As a physical entity the mobile clients perform the following tasks:

    1. Register with the sensor as soon as the application launches.

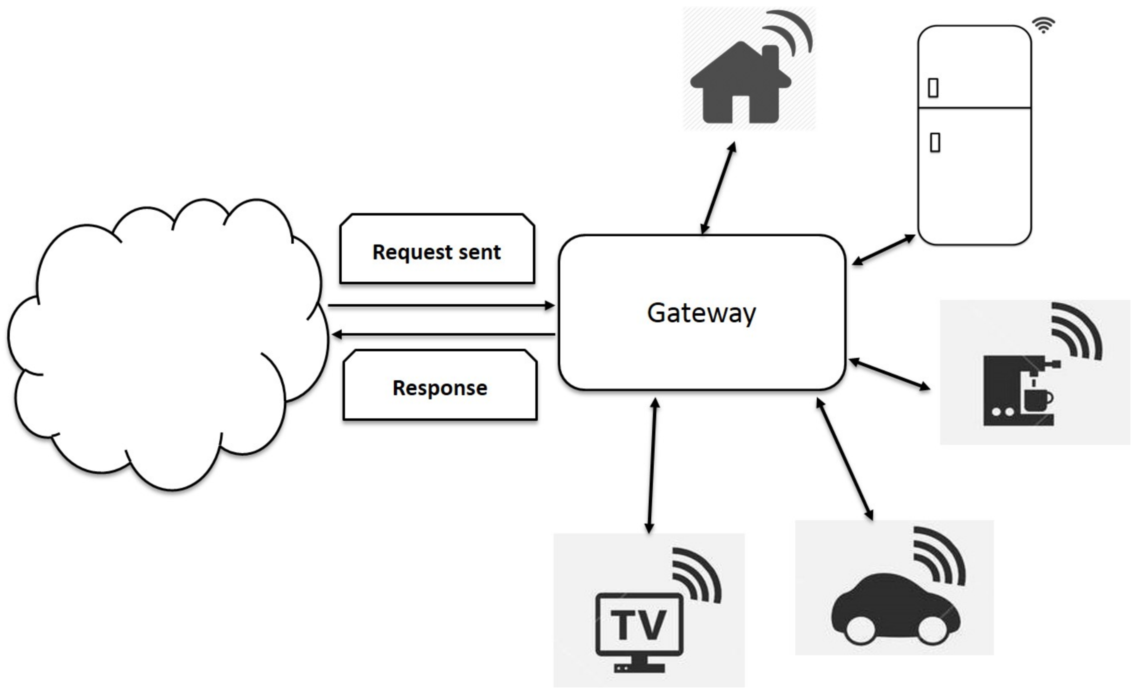


**Figure 4.8:** Cloud Application Interaction with the Gateway

2. Gather sensor data periodically.
3. Initiate web authorization requests.
4. Send the request along with context values using network packages.
5. Receive the web response and show it to the user as the data suggests. That means if it is an error report, it will show that in a dialog or if it is valid data, the device will show as it is supposed to show.

- **Cloud**: Cloud acts as a server and server receives requests from mobile clients with the calculated trust value and context values. They provide direct services to the mobile clients as well as forwarding the web requests to the IoT nodes. Rather than the requests going through the firewall or the enterprise server bus (ESB) which normally has authorization and authentication process for users to identify themselves, they are authorized by the cloud itself. The requests are rejected if the trust value is lower than the trust threshold that is initially set in the cloud. The request might not be rejected all the time. It has the capability to ask for further verification.

- **Gateway**: The gateway sends the connection request along with the list of available sensors to the cloud. So the cloud knows which gateway to pick if any request comes for that sensor. Cloud forwards the request to that gateway. The gateway receives the request from the cloud and creates a CoAP message with that request for the requested IoT node. The gateway is the CoAP server, and it sends the message to the IoT node. Whenever the response reaches from the IoT node, the gateway immediately sends it back to the cloud.

- **IoT Nodes**: Each IoT node has its functionality and also acts as a CoAP client. The nodes will be only connected to the gateway as the nodes only trust the gateway server. So whenever they receive a CoAP message from the gateway, it performs the action and sends an appropriate CoAP response (ACK, NON, RST) to the gateway.

### 4.2.2 Logical Architecture

From the logical perspective, the architectural structure is divided into four components based on functionality: the mobile devices, the cloud application, and the end nodes or sensors.

- **The Mobile/Portable Devices**: This component can be Smartphones, tablets, and PDAs, are the same as the mobile device component in the physical aspect. Portable devices are small, and that is why vulnerable to the open environment and can be compromised quickly. That is why it has more priority than others.

- **The Cloud Application**: The cloud application does not provide direct services. First, it analyzes the web request for authorization with contextual information and calculates trust value according to the trust policies. Based on the trust value and the trust threshold of this type of requests, the cloud application either accepts the request or rejects the request primarily and asks for further validation. The cloud application also saves the bad requests. This information will help administrators to understand the trends of the threats and how to define policies more effectively.

- **Gateway**: The Gateway acts as the doorkeeper for the IoT nodes. The gateway will have the information of all of the connected IoT nodes and then it sends a connection request to the cloud. The connection is secured as web socket is used. The gateway connects to the cloud and later sends the list of all available sensors. Whenever a request comes from the cloud, it converts the request to a CoAP message as the gateway is connected to all of the sensors via CoAP server-client protocol. So Gateway, as a server, sends the CoAP message to the specific IoT node and sends back the response received from the IoT node to the cloud.

- **The End Nodes (Sensors)**: The low powered Internet-enabled devices can accept CoAP requests from the gateway and can respond back to the gateway. The nodes are only allowed to communicate if the request is coming from the gateway. The IoT nodes interpret GET method as READ, POST method as WRITE, PUT method as CONTROL and RST as RESET configuration file.

## 4.3 Data Format and Flow

The Smartphone initiates the task by sending HTTPS requests in the cloud application. The content of the web request contains query parameters and context information in JSON (JavaScript Object Notation)

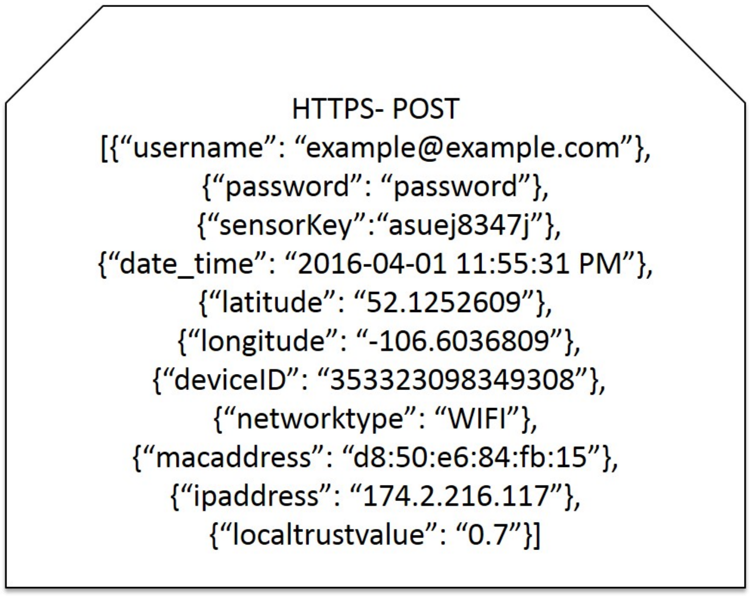

HTTPS- POST
[{“username”: “example@example.com”},
{“password”: “password”},
{“sensorKey”:“asuej8347j”},
{“date_time”: “2016-04-01 11:55:31 PM”},
{“latitude”: “52.1252609”},
{“longitude”: “-106.6036809”},
{“deviceID”: “353323098349308”},
{“networktype”: “WIFI”},
{“macaddress”: “d8:50:e6:84:fb:15”},
{“ipaddress”: “174.2.216.117”},
{“localtrustvalue”: “0.7”}]

**Figure 4.9:** Data Format POST From Mobile to Cloud Application

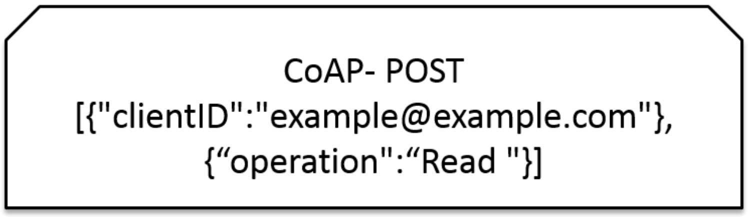
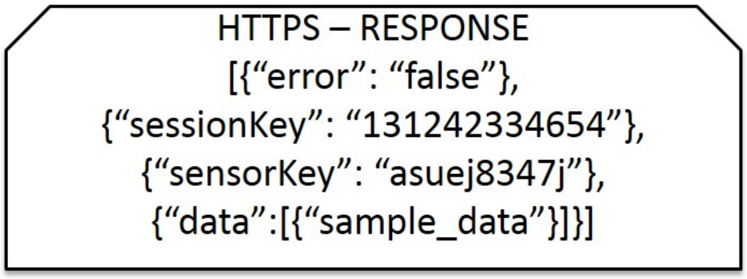
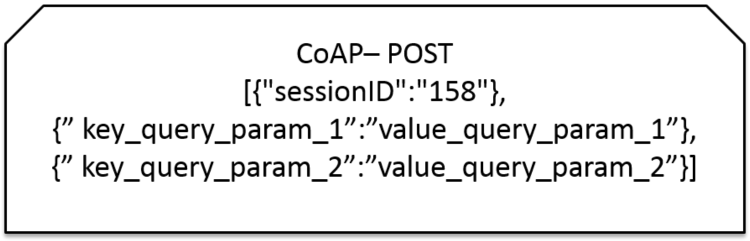

CoAP- POST
[{"clientID":"example@example.com"},
{“operation":“Read "}]

**Figure 4.10:** Data Format POST From Cloud Application to the Gateway



HTTPS – RESPONSE
[{“error”: “false”},
{“sessionKey”: “131242334654”},
{“sensorKey”: “asuej8347j”},
{“data”:[{“sample_data”}]}]

**Figure 4.11:** Data Format RESPONSE From Gateway to Cloud and then the Mobile Application



CoAP– POST
[{"sessionID":"158"},
{” key_query_param_1”:”value_query_param_1”},
{” key_query_param_2”:”value_query_param_2”}]

**Figure 4.12:** Data Format of further Query

format. The proposed architecture uses REST Framework to interchange data. Figure 4.9 shows the data in JSON format that transfers from the mobile to the cloud application. After receiving this request for authorization, the cloud application calculates the final trust value of the incoming request and either passes it to the mobile client with a session key or asks for further validation.

After approving the request, cloud sends a POST request to the the gateway in JSON format as shown in figure 4.10. Also as this is the session key that is just initiated, cloud also sends a response (similar to figure 4.10) back to the mobile device acknowledging that the request has been processed. Cloud application saves session ID in the database for later transactions. Each time the client sends any request with this session ID, the cloud application checks for validity. If the session is expired, the mobile client needs to verify again.

The gateway receives the JSON formatted web request from the cloud. The gateway parses and converts the request into a CoAP message. As the gateway uses CoAP architecture to communicate with the low-level sensors, it just converts same HTTPS request to the CoAP request with same data format (CoAP also uses REST Framework).

The IoT sensors reply back to the Gateway with the appropriate CoAP response (CON, ACK, NON, RST). Then the CoAP server converts that message back to web format and sends it back to the cloud as shown in figure 4.11. The same response sends back to the mobile application as well. The cloud application sends the stored session key to the mobile client. So for next transactions, the mobile application can use the session key while making requests as shown in figure 4.12.

## 4.4 System Functionalities

The system has the following step by step functionalities:

- The gateway device discovers and connects to the Sensors: The gateway device creates a CoAP server and initiates a broadcast request to all the clients to know about their status. These sensors can only accept requests from the gateway and send an ACK response to the gateway device. From the reply, the gateway will have information of all sensor devices.

- The gateway device connects with the Cloud: The gateway connects to the cloud by initiating a web socket response. The gateway has cloud's identity information and it sends the connection request to the cloud. The cloud accepts the request if the web socket request matches with the known gateway's identity information (MAC Address and IP Address). By applying this architecture, the system will provide a secure connection so that no external malware can break into the gateway [160]. The gateway will also send the information of active sensors. Each sensor will have a specific key.

  These two steps must happen anytime before the cloud tries to send the request to the cloud.

- **Gather Mobile Context Values:** An Android tablet is used as a portable client in the implementation. Whenever a user wants to know any sensor information, the mobile device or tablet also catches the device location, network configuration, and Internet connection information. A background service is to collect all the necessary information so that it can avoid delay in gathering device's context values. Table 4.1 represents data collected from Geolocation, Internet connectivity, and Mobile Network.

  **Table 4.1:** Collected Context Information

  | Geolocation | Internet Information | Time |
  |---|---|---|
  | Latitude | Mac Address | DateTime |
  | Longitude | Network Type | |

- **Calculating Local Trust Value:** The proposed system is now able to calculate its initial trust value based on the context information collected. The process starts with trust value-initialize to zero (0). Now at the very beginning, if the device location is not found, the whole process fails, and the request does not proceed. If the device's current location is found, the current trust value will be 50. After that, the background service will look for internet connectivity of the device and system date time. The background service tries to collect all the necessary information mentioned in Table 4.1.

  When collecting all the information is completed, then the application enters into the trust calculation part. Each context has its associated values and in this thesis, we only incorporate them together. So the local trust value will be the summation of retrieved context values. The minimum threshold value will be defined by the administrator while setting up the system. Also, this threshold value can be modified.

- **Invoking Web Services:** After computing the local trust value, the mobile application sends a request to the cloud. By sending the request, the application will wait for the response from the cloud. The location information, MAC and IP address, time, local trust value and the query string is sent to the cloud. The proposed architecture sends a web request along with query data in JSON (JavaScript Object Notation) format.

- **Analyzing Context Information and Policies:** The cloud receives all parameters from mobile clients. A set of defined policies needs to be in the configuration or policy file. Before starting calculation of *final trust value*, the cloud application retrieves those policies and matches with context information. The authentication can also be done in this phase or before sending to the cloud application. Those policies can be defined as precisely as they want.

  The mobile client is sending three (3) kinds of context information to the cloud as well as authentication information too. For the thesis, a simple set of policies of each kind is only considered. All of the trust policies will be stored in the policy file.

  The context values of a portable device to reveal the device's situation in many aspects, and those context values can be found through mobile sensors. For example, Motion sensors can indicate whether the users is driving, or walking or staying still. Also, it can show the rotation of the three axes. Environmental sensors can grab the temperature that helps to determine whether the device is currently outdoor or indoor. By having more context values, a more accurate situation can be determined. In future, the proposed architecture will be considered with more context values. Different sensors might need to have different level of security requirements. So threshold values and policies for each sensor can be set differently.

  When the cloud server receives the web requests along with the mobile device context information, they look up each corresponding trust policy. For instance, if the location is not home, but within the city of Saskatoon or near home, a trust value is given; if the location is not in Saskatoon, the trust value is given based on the policies declared. Trust values for each context category are summed up as the requests current trust value.

  The system administrator can set a list of allowed locations, Mac address lists, etc. If the current context falls on the approved list, the request gains more trust. If it falls on the rejected list, it loses all the trust value. If the context value does not fit into accepted or rejected list, still it will have a trust value. Whenever a context value does not fall into either accepted or rejected list, the trust module keeps track of this kind of suspicions. If the number of suspicions increases and crosses threshold level defined by the administrator, the request is sent to the user for the verification code.

- **Calculating Final Trust Value in the Cloud:** Different approaches can be followed to measure trust value of a mobile device. In this thesis, the values are matched with policies and each value is assigned based on the matching result. If it matches, then the trust value adds the highest value. If it does not match and does not fall into restricted list, then it has some value. The context values are matched in this way, and if the final value exceeds the minimum threshold, then the device has the permission to access IoT end nodes. However, if it fails to gain enough trust, then it will be asked for further validation. Each policy might have different weight factor. However, in this thesis, all policies were considered as of same importance.

- **Forward Web requests to the Gateway:** Cloud application forwards this web request to the gateway. The gateway is already connected to the cloud through a web socket. The gateway server receives the request from the cloud and creates an appropriate CoAP message for the sensors. The sensors are working as CoAP client, so it receives the message from the gateway server and acts accordingly. After that, it also sends an appropriate response(CON if the task is completed successfully, NON if the task is not completed successfully) to the gateway server. The gateway will transform the CoAP message to an HTTPS request and sends it back to the cloud.

- **Response back to the mobile client:** After receiving the response back from the gateway server, the cloud sends the response back to the mobile client. The cloud application saves the session key to verify further transactions.

- **Mobile Client makes further requests:** After receiving the response from the cloud application with session key the mobile application can understand that the authorization part is done and the IoT sensors are ready to interact. Now the mobile application can make the necessary request to READ/WRITE or CONTROL these IoT sensors. The cloud application at first checks whether the request contains a valid session ID. If yes, then it sends the request to the desired IoT node through the gateway server and the node then can deliver the response. If the session ID is no longer valid such as session expired, then the cloud application rejects the request and send back to the mobile client for re-authorization. The proposed architecture works as mentioned above.

## 4.5 Summary

A bi-directional authorization system is described with physical and logical architecture as well as a data format. This authorization system can explain all of the features that have been stated in the research goal section 2.1. During authorization, Dynamism feature of attribute based access control has been applied to protect users from known and unknown threats. Unwanted requests are validated with users' contextual information. The proposed system cuts off untrusted requests initially from the mobile application and later on compares with the defined policies. This two state cut off approach gives the resources a huge security comfort.

# Chapter 5

# Performance Evaluation

This chapter describes the proposed experiments to measure the response time of the proposed architecture. Chapter 4 discusses the proposed system to minimize the unauthorized web requests, and this chapter focuses on measuring the performance of accessing cloud-centric IoT services and resources. As these interactions are happening among multiple hardware devices, the importance of calculating and comparing the communication cost, and system overhead become the necessity. The evaluation is performed separately for each step and then the overall interaction of the proposed architecture.

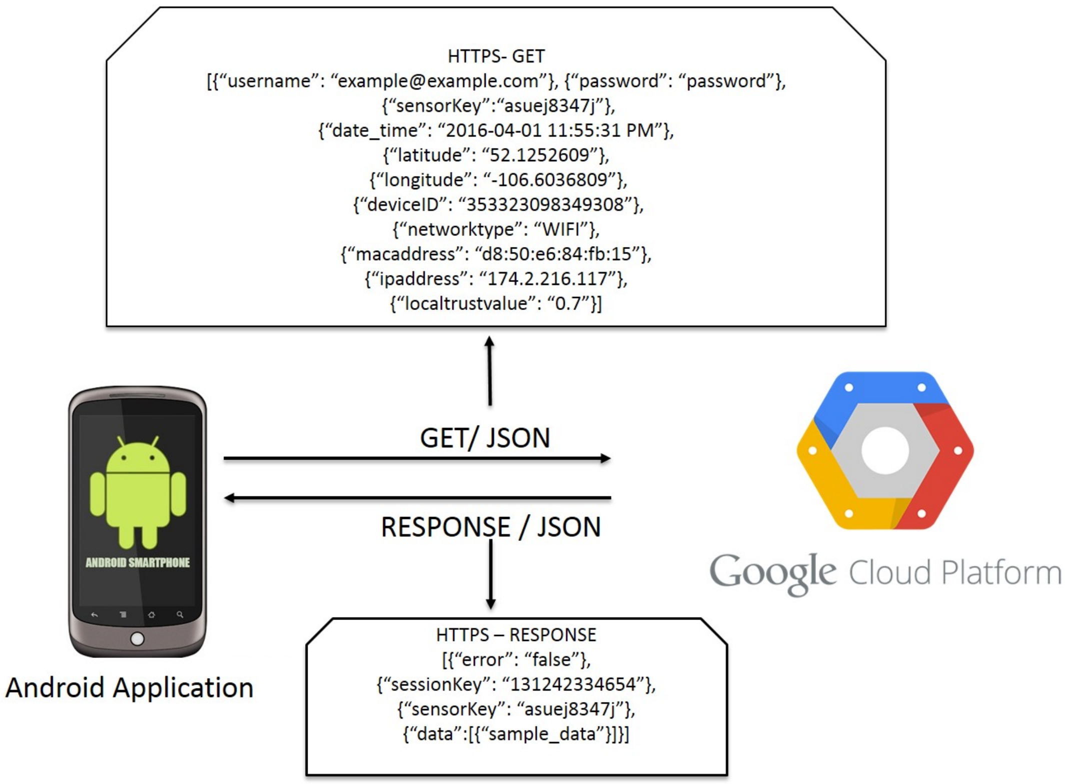
## 5.1 Implementation Details and Experimental Setup

The experiments are simulated to provide the observer a reliable, and fast control over the IoT environment. The experiments are designed to prove that the proposed architecture is implementable, flexible, and uses low system overhead and takes less interaction time. The details of implementation and experimental setup is defined below:

### 5.1.1 Mobile Clients

An Android application is developed using Java 1.7 SDK and Android API 22 Platform to perform as a mobile client. The Android application collects context information from sensors and evaluates the initial trust value. It also generates a request and sends the requests to the cloud using the Java Restlet 2.0 client package.An Asus Google Nexus 7 tablet, running Android OS 6.0.1, is used to deploy the Android application The detailed device specifications of Android tablet are listed below:

- CPU: Quad-core 1.5 GHz Krait.
- Memory: 2 GB.
- Storage: 16 GB.
- Sensors: GPS, Accelerometer, Gyro, Barometer, Compass, and Others.



```
HTTPS- GET
[{"username": "example@example.com"}, {"password": "password"},
{"sensorKey":"asuej8347j"},
{"date_time": "2016-04-01 11:55:31 PM"},
{"latitude": "52.1252609"},
{"longitude": "-106.6036809"},
{"deviceID": "353323098349308"},
{"networktype": "WIFI"},
{"macaddress": "d8:50:e6:84:fb:15"},
{"ipaddress": "174.2.216.117"},
{"localtrustvalue": "0.7"}]
```

```
HTTPS – RESPONSE
[{"error": "false"},
{"sessionKey": "131242334654"},
{"sensorKey": "asuej8347j"},
{"data":[{"sample_data"}]}]
```

**Figure 5.1:** The scenario of Experiment 1

### 5.1.2 The Gateway Server

In this thesis, Raspberry PI 3, Intel Edison, and MACBOOK Pro are used as gateway servers. Raspberry PI 3 is used for most of the experiments. The same gateway server is implemented on both Intel Edison and MACBOOK Pro to compare the scalability and reliability of Raspberry PI. The detailed specification of these devices are given below:

- CanaKit Raspberry PI 3 is used to simulate IoT environment and deploy gateway server. This Raspberry PI 3 includes an ARMv8 quad-core Cortex-A53 CPU with 1.2 GHz processing speed and 1GB RAM. It also has built-in Bluetooth and WIFI port as well as 10/100 Ethernet port. The gateway server and both CoAP server and client applications are implemented in Go programming language. Go binary distribution 1.6.3 is used to setup Go environment. Both the gateway server and the CoAP server are deployed in one Raspberry PI 3. The sensor nodes are referred as the CoAP clients which are implemented in different applications and also deployed in another Raspberry PI 3. The CoAP server and gateway server listen to different ports for incoming requests.

- Intel Edison is a similar Internet-enabled low powered device as Raspberry PI. It combines with sensors and provides both hardware and software platform to enable the opportunity to invent new IoT products and solutions. The Intel Edison Board has the following specifications:

1. CPU: Intel Atom dual-core processor at 500 MHz.
2. Memory: 1 GB DDR3 RAM
3. Open source software development environment.
4. Cloud Adaptability.

- MACBOOK Pro is used as the gateway server to test how well the gateway server performs when a high powered device is used. The MACBOOK Pro has the following specifications:

  1. OS: OS X Yosemite.
  2. CPU: 2.7 GHz Intel Core i7.
  3. Memory: 16 GB 1600 MHz DDR3.
  4. Wide range of development environment.

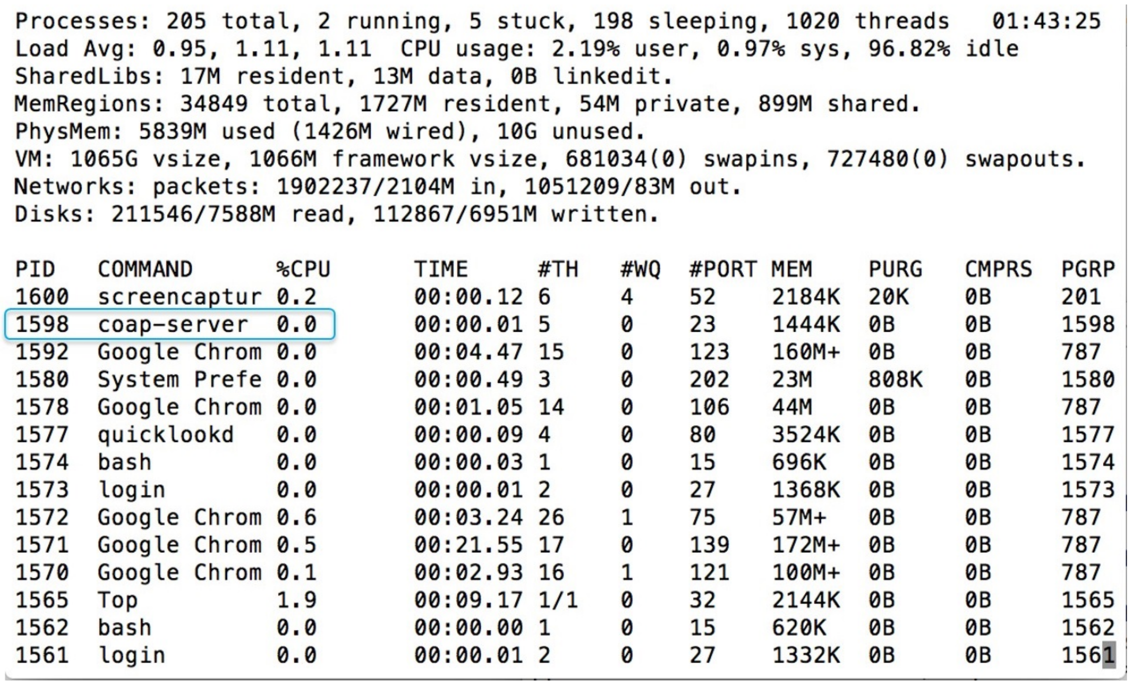
### 5.1.3 The Cloud

Lastly, A cloud-based web application is developed to evaluate context information and to make access decisions. The cloud application holds the logic to evaluate a web request. Whenever the cloud receives a GET request from a mobile client, it evaluates the request based on the context information with the predefined policies and sends an appropriate response back to that mobile client. However, if it gets a POST/PUT/DELETE request, then it analyzes and later forwards the request to the gateway server. The cloud application is hosted on Google App Engine. Google App Engine is highly scalable, easy to implement, and cost effective. Java Restlet API 2.0 is used to develop the cloud application. Java Restlet API follows REST architecture style and is helpful to develop lightweight solutions.

## 5.2 Performance Matices

The total response time, as well as mobile-cloud interaction time and CoAP-cloud interaction time for varying number of policy entities, are measured and compared with the results of the existing architecture [5] to evaluate the robustness of the proposed system in a real environment. The performance matrices for the evaluation are

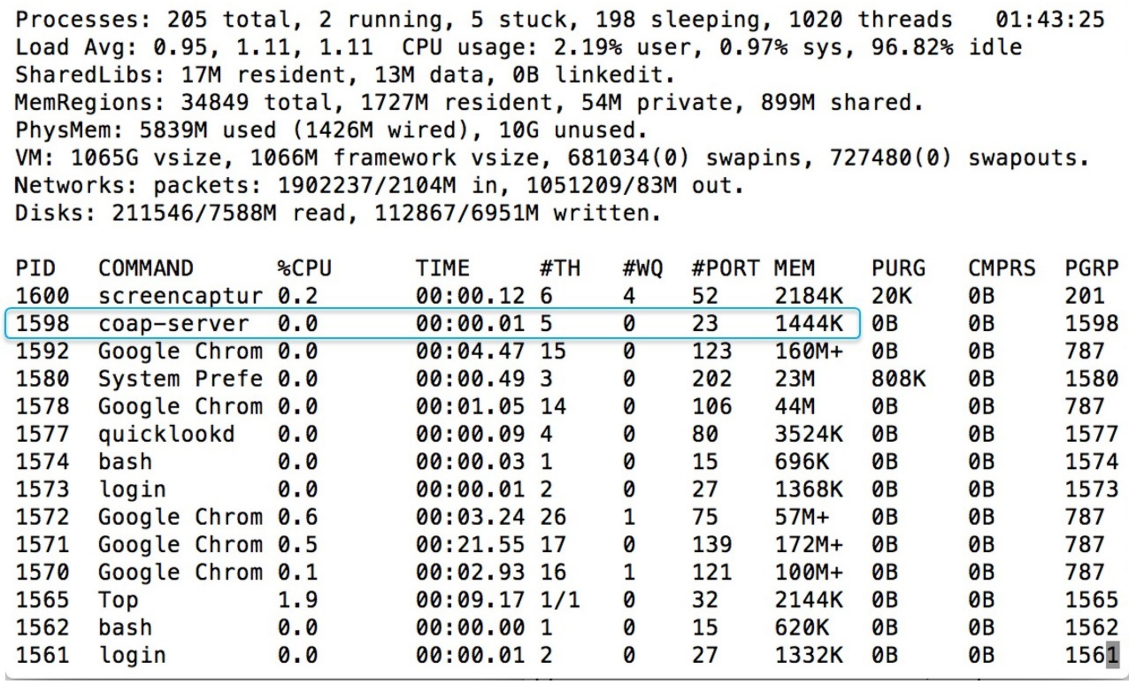
- Mobile-Cloud Interaction Time: It is the communication time between the Android application and Google cloud service.
- CoAP-Cloud Interaction Time: It defines the communication time between the cloud service and the sensor nodes which are referred as the CoAP clients.
- CPU Usage: The CPU usage occurred during the request processing in the gateway server is measured for Raspberry PI 3, Intel Edison, and MACBOOK Pro. Both Synchronous and asynchronous requests



```
Processes: 205 total, 2 running, 5 stuck, 198 sleeping, 1020 threads   01:43:25
Load Avg: 0.95, 1.11, 1.11  CPU usage: 2.19% user, 0.97% sys, 96.82% idle
SharedLibs: 17M resident, 13M data, 0B linkedit.
MemRegions: 34849 total, 1727M resident, 54M private, 899M shared.
PhysMem: 5839M used (1426M wired), 10G unused.
VM: 1065G vsize, 1066M framework vsize, 681034(0) swapins, 727480(0) swapouts.
Networks: packets: 1902237/2104M in, 1051209/83M out.
Disks: 211546/7588M read, 112867/6951M written.

PID   COMMAND      %CPU      TIME     #TH    #WQ  #PORT MEM    PURG   CMPRS  PGRP
1600  screencaptur 0.2       00:00.12 6      4    52    2184K  20K    0B     201
1598  coap-server  0.0       00:00.01 5      0    23    1444K  0B     0B     1598
1592  Google Chrom 0.0       00:04.47 15     0    123   160M+  0B     0B     787
1580  System Prefe 0.0       00:00.49 3      0    202   23M    808K   0B     1580
1578  Google Chrom 0.0       00:01.05 14     0    106   44M    0B     0B     787
1577  quicklookd   0.0       00:00.09 4      0    80    3524K  0B     0B     1577
1574  bash         0.0       00:00.03 1      0    15    696K   0B     0B     1574
1573  login        0.0       00:00.01 2      0    27    1368K  0B     0B     1573
1572  Google Chrom 0.6       00:03.24 26     1    75    57M+   0B     0B     787
1571  Google Chrom 0.5       00:21.55 17     0    139   172M+  0B     0B     787
1570  Google Chrom 0.1       00:02.93 16     1    121   100M+  0B     0B     787
1565  Top          1.9       00:09.17 1/1    0    32    2144K  0B     0B     1565
1562  bash         0.0       00:00.00 1      0    15    620K   0B     0B     1562
1561  login        0.0       00:00.01 2      0    27    1332K  0B     0B     1561
```

**Figure 5.2:** Top command to measure CPU Usage.



```
Processes: 205 total, 2 running, 5 stuck, 198 sleeping, 1020 threads   01:43:25
Load Avg: 0.95, 1.11, 1.11  CPU usage: 2.19% user, 0.97% sys, 96.82% idle
SharedLibs: 17M resident, 13M data, 0B linkedit.
MemRegions: 34849 total, 1727M resident, 54M private, 899M shared.
PhysMem: 5839M used (1426M wired), 10G unused.
VM: 1065G vsize, 1066M framework vsize, 681034(0) swapins, 727480(0) swapouts.
Networks: packets: 1902237/2104M in, 1051209/83M out.
Disks: 211546/7588M read, 112867/6951M written.

PID   COMMAND      %CPU      TIME     #TH    #WQ  #PORT MEM    PURG   CMPRS  PGRP
1600  screencaptur 0.2       00:00.12 6      4    52    2184K  20K    0B     201
1598  coap-server  0.0       00:00.01 5      0    23    1444K  0B     0B     1598
1592  Google Chrom 0.0       00:04.47 15     0    123   160M+  0B     0B     787
1580  System Prefe 0.0       00:00.49 3      0    202   23M    808K   0B     1580
1578  Google Chrom 0.0       00:01.05 14     0    106   44M    0B     0B     787
1577  quicklookd   0.0       00:00.09 4      0    80    3524K  0B     0B     1577
1574  bash         0.0       00:00.03 1      0    15    696K   0B     0B     1574
1573  login        0.0       00:00.01 2      0    27    1368K  0B     0B     1573
1572  Google Chrom 0.6       00:03.24 26     1    75    57M+   0B     0B     787
1571  Google Chrom 0.5       00:21.55 17     0    139   172M+  0B     0B     787
1570  Google Chrom 0.1       00:02.93 16     1    121   100M+  0B     0B     787
1565  Top          1.9       00:09.17 1/1    0    32    2144K  0B     0B     1565
1562  bash         0.0       00:00.00 1      0    15    620K   0B     0B     1562
1561  login        0.0       00:00.01 2      0    27    1332K  0B     0B     1561
```

**Figure 5.3:** Top command to measure Memory Usage.

are generated, and CPU usage is measured for both cases. To compute the CPU usage, *top* command is used. The *top* command runs on all devices and provides a dynamic real-time view of the running system. The CoAP-server is highlighted in the figure 5.2. This *top* command works on all three platforms. Later, the CPU usage in the cloud is also measured by varying both number of requests and number of policy entities.

- Memory Usage: The memory usage in all three systems are measured when these devices act as a Gateway server. Both Synchronous and asynchronous requests are generated, and memory usage is measured for both cases. To compute the memory usage, *top* command is used in here also. The *top* command runs on all devices and provides a dynamic real-time view of the used memory of all the running system (see figure 5.3). The *top* command in MACBOOK Pro gives memory in KB metrics. The *top* command in both Intel Edison and Raspberry PI give memory in percentile format. However, the *free -k* command helps to figure out the memory metrics used in the system (see figure 5.4). Later, the Memory overhead of cloud application in the cloud is also measured by varying both number of requests and number of policy entities.

- System Response Time: It is the total round trip time(RTT) of the system which is the total communication time from the mobile client to CoAP client and again from CoAP client to mobile client.

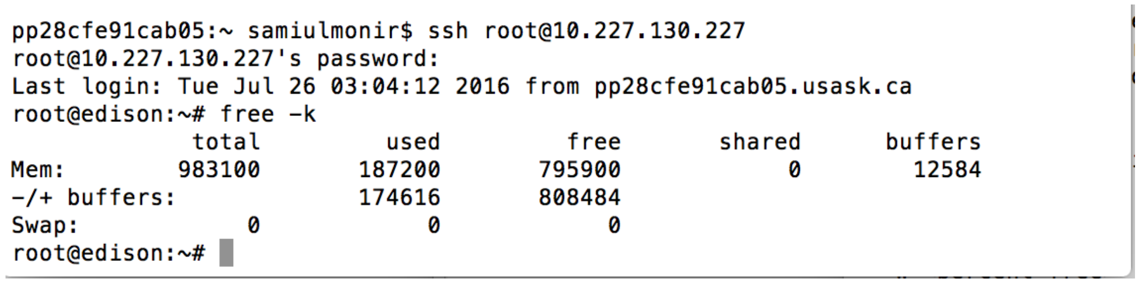
- System Response Time: It is the total round trip time(RTT) of the system which is the total communication time from the mobile client to CoAP client and again from CoAP client to mobile client for different types of requests (GET, POST, and PUT).

## 5.3 Experiment Results

First three matrices are evaluated for varying policy entities. The policy entities are expressed as rules. Each sensor has its individual characteristics and can have own policies to approve the access. Suppose to turn off a light bulb, and the time needs to be after 8 PM and location need to be inside the building. Now System Administrator can build the rules based on that, and that is considered as one of the policy entity. Each sensor may have multiple rules to define the access privilege. The later experiments are conducted to proof the benefit of using Raspberry PI. Lastly, the overall system responses for different request types are recorded.

### 5.3.1 Mobile-Cloud Interaction Time:

In the first experiment, the communication time between the Android application and Google cloud service is measured. The communication time is calculated by measuring the response time against the policy entities. The cloud application holds a configuration file with 10, 100, 1000, 10000 policy entities. These policy entities



```
pp28cfe91cab05:~ samiulmonir$ ssh root@10.227.130.227
root@10.227.130.227's password:
Last login: Tue Jul 26 03:04:12 2016 from pp28cfe91cab05.usask.ca
root@edison:~# free -k
             total       used       free     shared    buffers
Mem:        983100     187200     795900          0      12584
-/+ buffers:           174616     808484
Swap:            0          0          0
root@edison:~# 
```

**Figure 5.4:** Free command

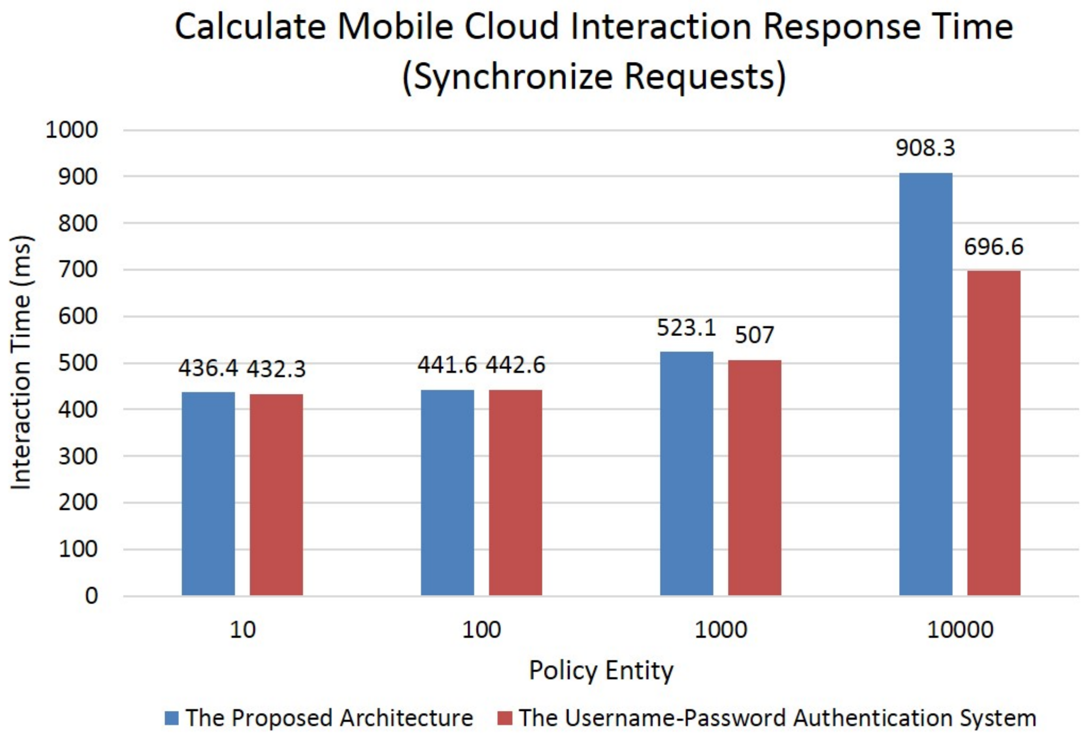


**Figure 5.5:** Calculate Mobile Cloud Interaction Response Time (Synchronize Requests).

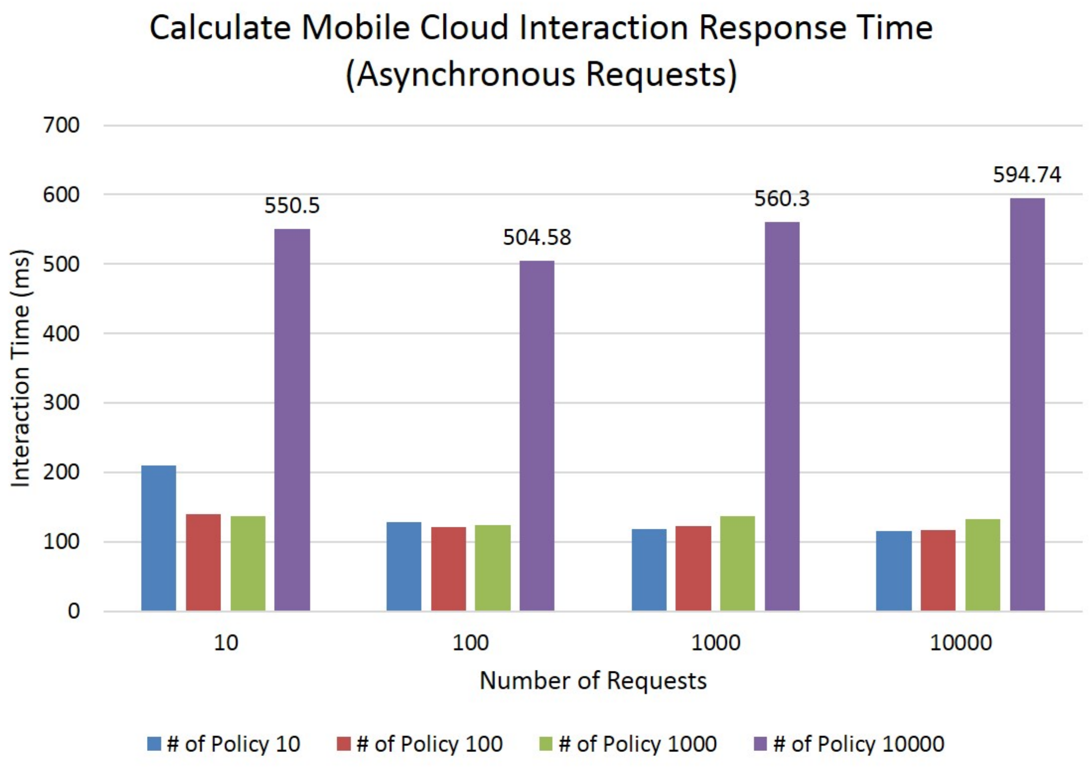


**Figure 5.6:** Calculate Mobile Cloud Interaction Response Time (Asynchronous Requests).

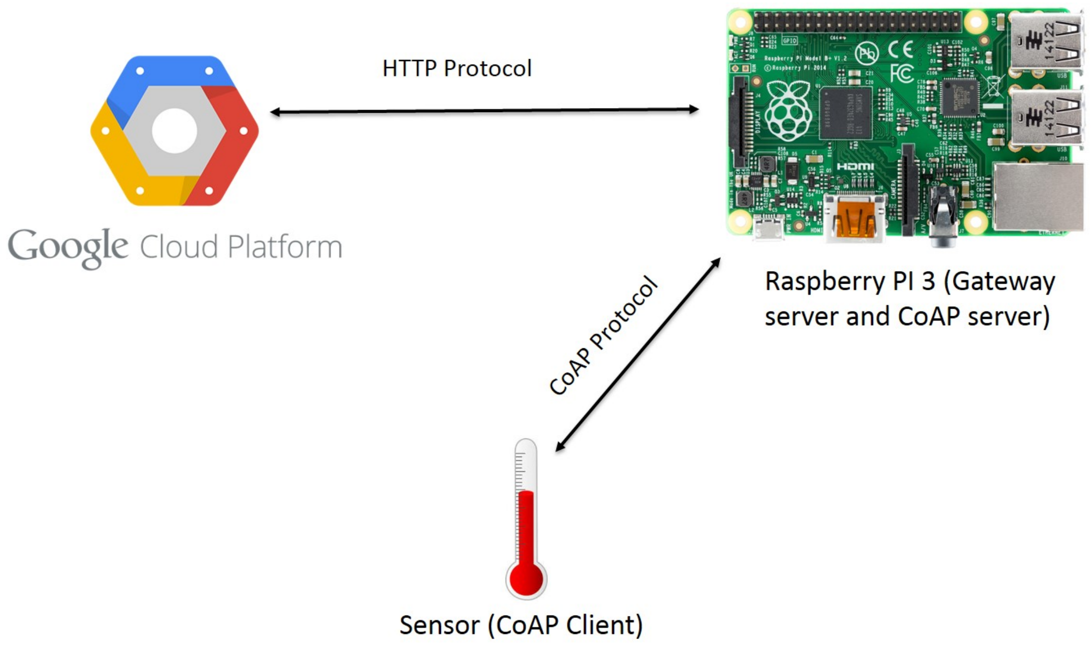


**Figure 5.7:** The experiment scenario of CoAP-cloud Interaction

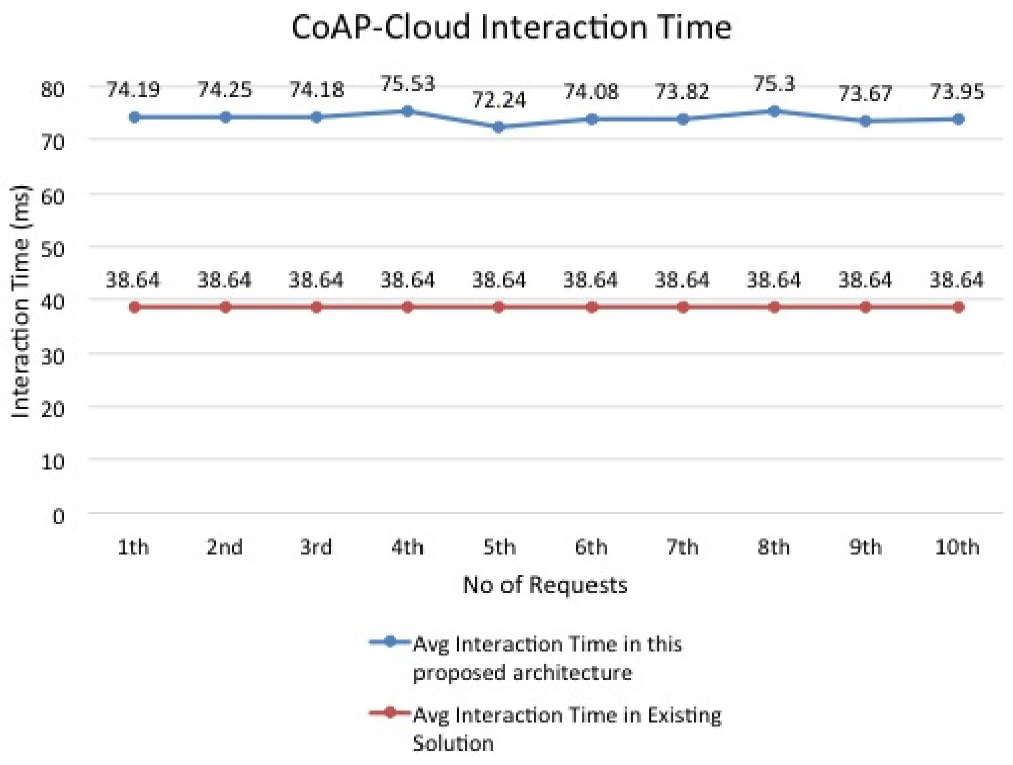


**Figure 5.8:** Calculate CoAP and Cloud Interaction Response Time. The red line represents the response time of CoAP interaction described in [5]

represent the values that are permitted to access the sensor. Each sensor might have multiple entities. The Android application sends the context values to evaluate against these policy entities. The mobile-cloud interaction time of the proposed architecture is measured against the Username-Password authentication system. The figure 5.1 shows the scenario of the first experiment.

1. **Synchronize Requests**: Figure 5.5 shows the comparison between the proposed architecture and the Username-Password authentication system in terms of mobile-cloud interaction time for varying sets (10,100,1000,10000) of policy entities. The X-axis represents the number of policy entity, and the Y-axis represents the interaction time in milliseconds. The main focus is to understand the increased delay due to the added trust value calculation in the cloud. For each request, the mobile application is restarted to include the system overload. The time is computed from the start of calculation of local trust value to receive the response from the cloud. In the cloud, the number of policies is increased to observe how that affects the response time. The results show that the time of reply does not vary much for 10 to 100 policy entities. Although the response time does not vary much till 1000, for a large number of entities (10000), the response time increases by 39% as the cloud needs more time to search through the entities and find out the exact matching. Also, it is highly unlikely that the system might have that many rules (10000) for a small office or home automation system.

2. **Asynchronous Requests**: Figure 5.6 shows the comparison of number of requests in terms of mobile-cloud interaction time for varying sets (10,100,1000,10000) of policy entities. The X-axis represents the number of requests sent, and the Y-axis represents the interaction time in milliseconds. The requests are sent asynchronously at a varying rate of 10, 100, 1000, and 10000 with a random interval. The start time begins the moment the first request is started and the time ends when the last request is processed and then divided by the number of requests to get the average interaction time. In this experiment, the time of reply does not vary much for 10 to 1000 policy entities for 10 to 1000 requests. When the policy entity reaches 10000, irrespective of the number of requests, the interaction time takes more time to complete.

### 5.3.2 CoAP-Cloud Interaction Time:

In the second experiment, the communication time between the cloud service and sensor nodes (CoAP clients) is measured. The Raspberry PI initiates HTTP server and CoAP server and then listens to different ports for incoming requests. The CoAP-Cloud interaction time is measured from when CoAP client sends a request to get back the response from the CoAP server. The CoAP server receives incoming CoAP request from CoAP client and converts the message to HTTP request. Afterward, the CoAP server sends the message to the cloud and replies back to the client CoAP application.

The figure 5.7 shows the experiment scenario of CoAP-Cloud interaction. The main focus is to understand the increased communication cost due to the addition of cloud interaction from Raspberry PI. Figure 5.8

showed CoAP-Client interaction time of the proposed IoT environment from 1st to the tenth request and compared with the existing proposed architecture [5]. The interaction time includes Raspberry PI and sensor interaction. The CoAP client sends a CoAP message to the gateway server. The gateway server receives the CoAP message and sends to the cloud through HTTP POST protocol. The HTTP server receives the response from the cloud and replies back to the CoAP client for confirmation. The time needs to complete an aforementioned full task is considered as CoAP-Cloud interaction time. In this experiment, ten requests have been sent to predict the average duration. The average time required to complete the task proposed in this architecture is $74 : 12ms$. 38.64 is required to complete the existing approach proposed in [5], and that is why only a straight line is drawn for this graph to compare with the proposed architecture. The results show that the CoAP interaction time is not time-consuming compared to the existing proposed architecture [5] and is reliable. The existing proposed architecture [5] calculates only the CoAP interaction whereas the proposed architecture in this thesis includes HTTP request-response as well.

### 5.3.3 CPU Usage:

The CPU overhead means the processing power is needed to perform a particular task. In the fifth experiment, the CPU usage is measured in all of the three platforms. These platforms have different specifications but same code, and same development environment is used. The requests are sent from CoAP client, and these requests are both synchronous and asynchronous. Later, the CPU overhead of cloud is also measured.

1. CoAP-Cloud Interaction CPU Usage: At first, the CoAP client sends 10, 100, 1000, and 10000 synchronize POST requests to the gateway server. The CoAP server running in the gateway server listens to these requests, transforms into an HTTP request, and sends to the cloud. The cloud replies back, and the gateway server answers back to the CoAP client. The CPU usage of whole interaction time is measured. Then the whole procedure is completed again with Intel Edison and MACBOOK Pro. Also, the same process is repeated with 10, 100, 1000, and 10000 asynchronous POST request.

   Figure 5.9 shows the CPU usage of gateway server application for synchronized requests. For 10 requests, the CPU usage value is too small for all devices, and that is why it has been omitted from the graph. The result shows that the CPU usage increases as the number of requests increases. For 1000 requests, the CPU usage of Raspberry PI increases 180% than MACBOOK Pro whereas Raspberry PI takes 31% less CPU usage compared to Intel Edison. So for synchronized requests, Raspberry PI has less CPU overhead than Intel Edison.

   Figure 5.10 shows the CPU usage of gateway server application for asynchronous requests. The CPU usage value is too small for ten requests, and that is why it has also been omitted from the graph. The result shows that the CPU usage increases as the number of requests increases. For 1000 requests, the CPU usage of Raspberry PI increases 157% than MACBOOK Pro whereas Raspberry PI takes 55% less CPU usage compared to Intel Edison. So for synchronized requests, Raspberry PI has less CPU

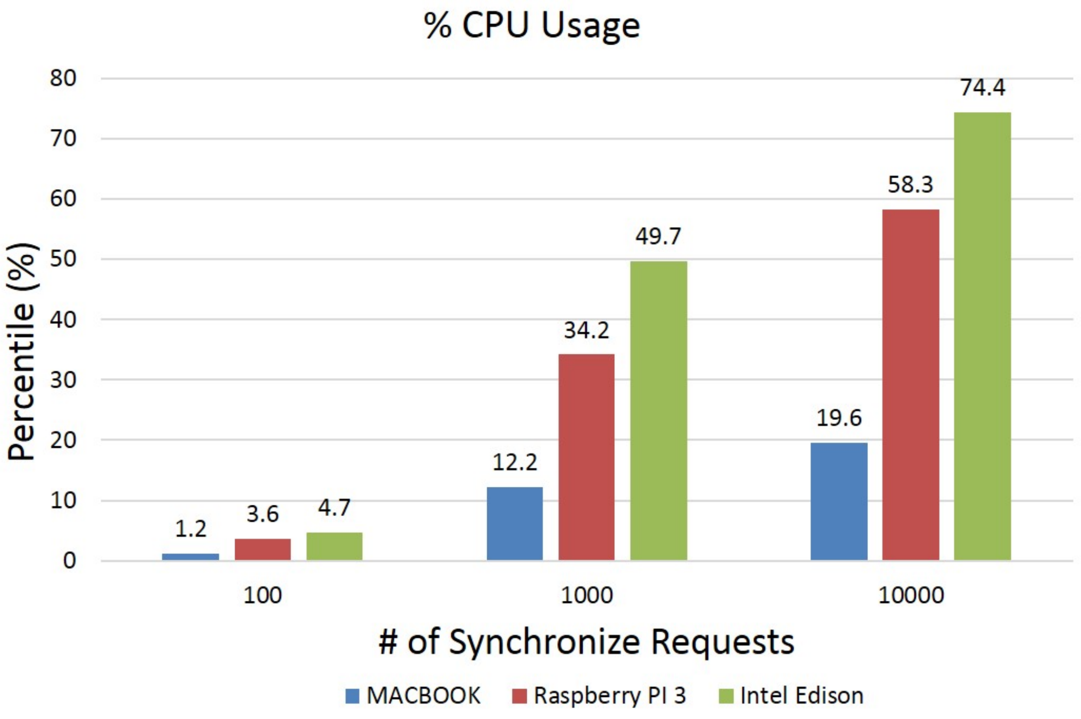


**Figure 5.9:** CPU Usage of Synchronize Requests between Gateway Server and Cloud.

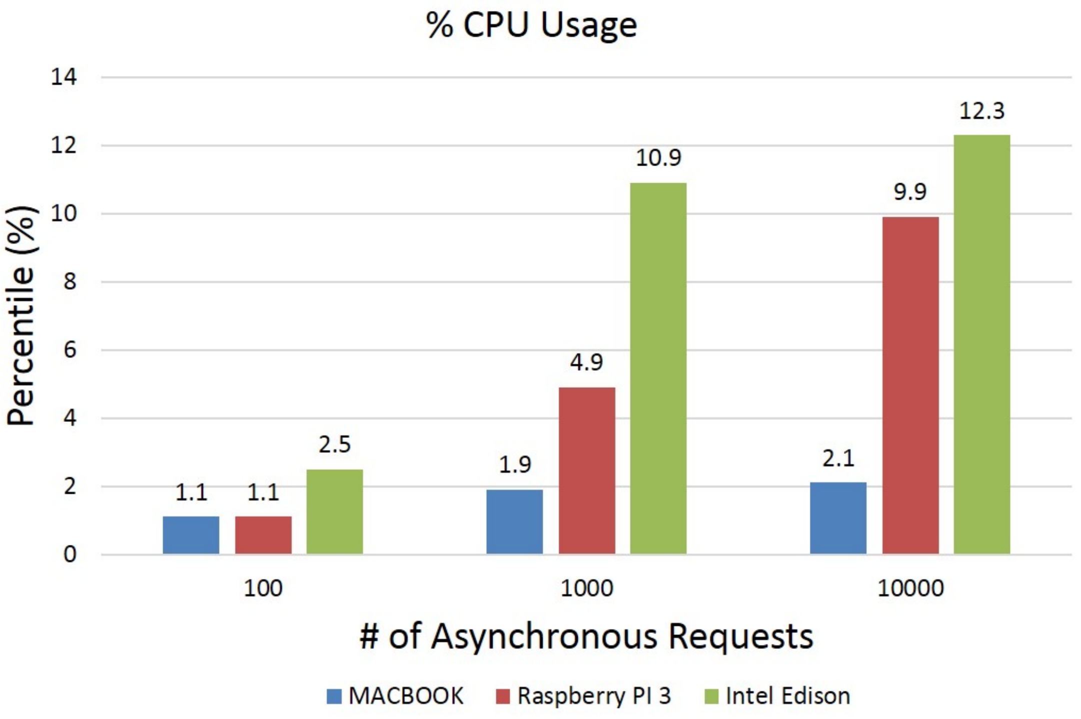


**Figure 5.10:** CPU Usage of Asynchronous Requests between Gateway Server and Cloud.

overhead than Intel Edison. Also, for asynchronous requests, the Raspberry PI is performing better than synchronized requests. For 10000 asynchronous requests, MACBOOK takes 2.1% CPU whereas Raspberry PI and Intel Edison needs more CPU cycles.

2. Mobile-Cloud Interaction CPU Usage: Figure 5.11 shows the CPU usage of the cloud application for the asynchronous requests. The X-axis represents the number of asynchronous requests send over the time, and the Y-axis represents the CPU time in (Cycles/sec). The primary focus is to understand the usage of the cloud CPU cycles due to the increase in the number of requests and policy entities. For 10 and 100 policy entities, the results show that the CPU usage is low for each set of requests. For 1000 policy entities, the CPU usage increases higher along with the number of requests. For 10000 policy entities and 10000 requests, the CPU usage increases much higher than other sets of requests.

### 5.3.4 Memory Usage:

The memory overhead means the memory is needed to perform a particular task. In the sixth experiment, the memory usage is measured in all of the three platforms. Similar to the measurement of CPU overhead, the requests are sent from CoAP client, and these requests are sent in separate batches using both synchronous and asynchronous manner.

1. CoAP-Cloud Interaction Memory Usage: At first, the CoAP client sends 10, 100, 1000, and 10000 synchronize POST requests to the gateway server. The CoAP server running in the gateway server listens to these requests, transforms into an HTTP request, and sends to the cloud. The cloud replies back, and the gateway server replies back to the CoAP client. The CPU usage of whole interaction time is measured. Then the whole procedure is completed again with Intel Edison and MACBOOK Pro. The same process is repeated with 10, 100, 1000, and 10000 asynchronous POST request.

   Figure 5.12 shows the memory usage of gateway server application for synchronized requests. For 10 requests, the memory usage is low for all of the devices. Approx 5 MB used by Intel Edison and this is the highest among three. The result shows that the memory usage increases as the number of requests increases. For 1000 requests, the memory usage of Raspberry PI increases by 17% than MACBOOK Pro whereas Raspberry PI takes 18% less memory usage compared to Intel Edison. The result also shows that the percentile difference (172% to 17%) of memory usage of MACBOOK Pro and Raspberry PI decreases rapidly when the number of requests increases from 100 to 1000. But from 1000 to 10000, the memory usage reaches up to 47%. So around 1000 synchronize requests, the optimum memory usage can be found. For 10000 synchronize requests, approx. 10 MB memory usage is needed for Intel Edison whereas 8.5 MB is needed for Raspberry PI.

   Figure 5.13 shows the memory usage of gateway server application for asynchronous requests. The memory usage value shows a bit different results in here. For a small number of requests such as 10, the

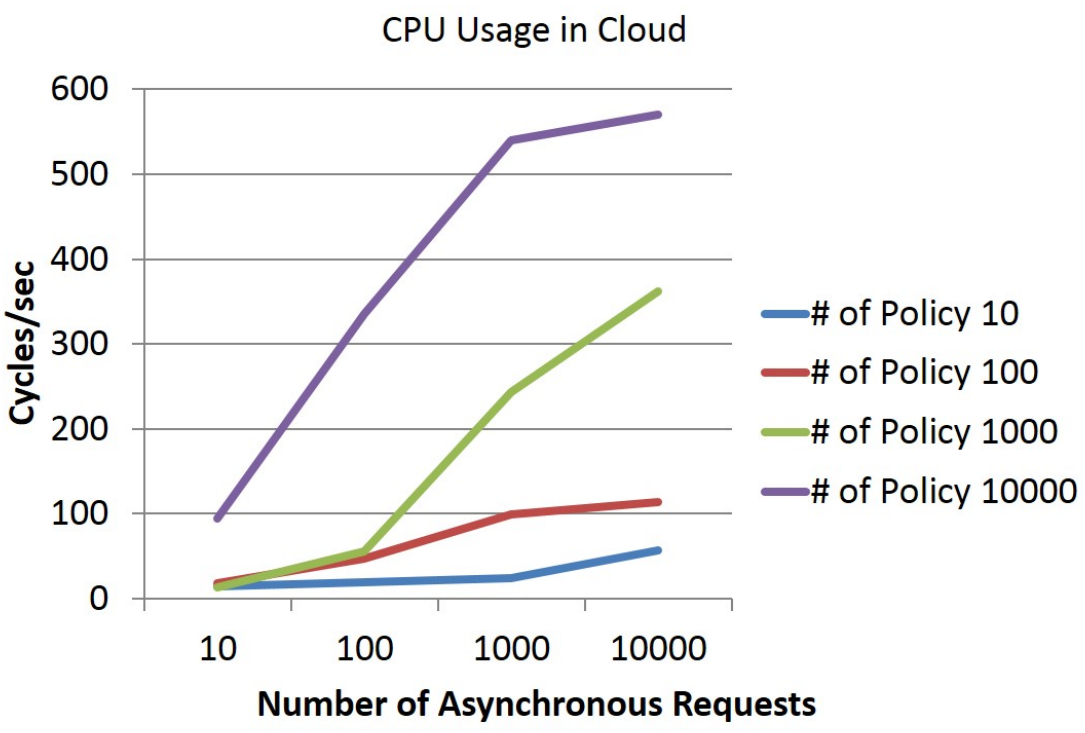


**Figure 5.11:** CPU Usage of Asynchronous Requests between Mobile Client and Cloud.

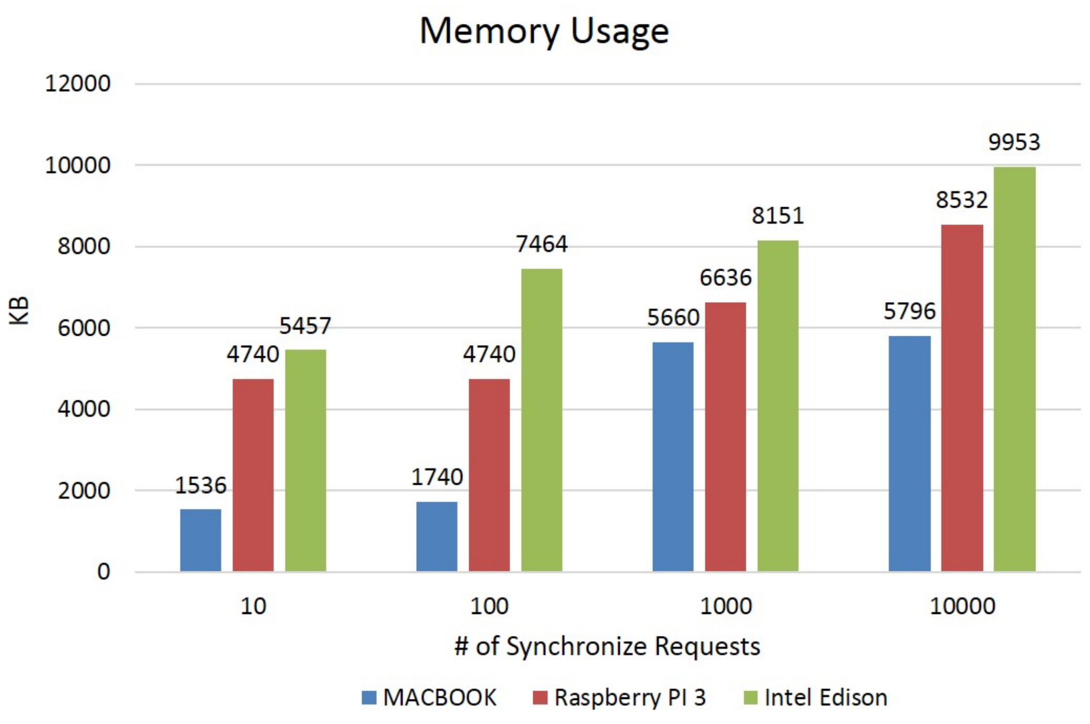


**Figure 5.12:** Memory Usage of Synchronize Requests between Gateway Server and Cloud.

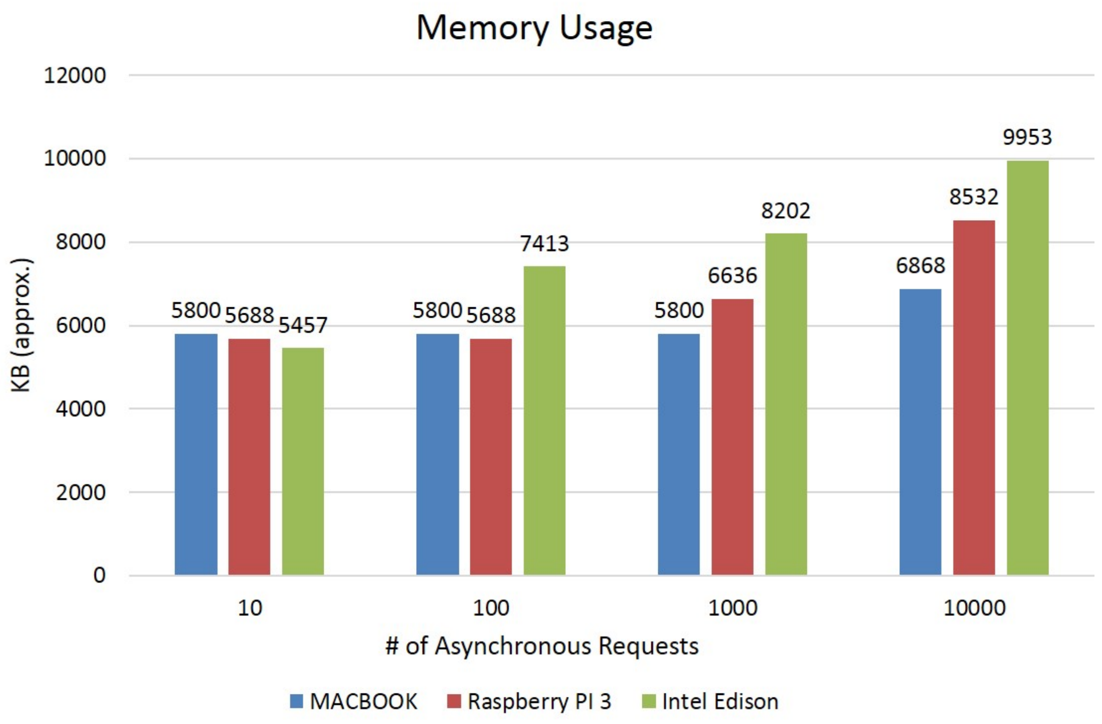


**Figure 5.13:** Memory Usage of Asynchronous Requests between Gateway Server and Cloud.

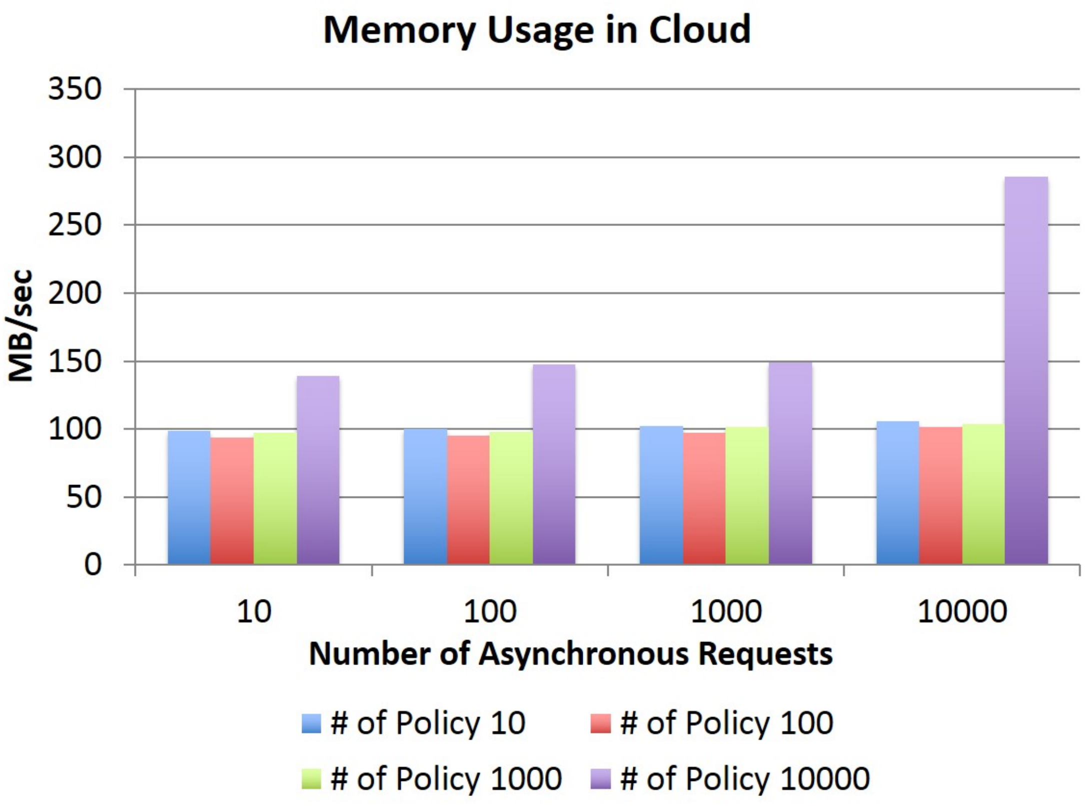


**Figure 5.14:** Memory Usage of Asynchronous Requests between Mobile Client and Cloud.

Intel Edison takes less memory to execute the gateway server than the other two. For 100, Intel Edison rises above all and Raspberry PI performs better. However, for 1000 requests, the memory usage of Raspberry PI increases only 14% than MACBOOK Pro whereas Raspberry PI takes 19% less memory usage compared to Intel Edison. So for the asynchronous requests, Raspberry PI has less memory overhead than Intel Edison when the number of asynchronous requests reaches minimum 100.

2. Mobile-Cloud Interaction Memory Usage: Figure 5.14 shows the memory usage of the cloud application for asynchronous requests. The X-axis represents the number of asynchronous requests sent over the time, and the Y-axis represents the Memory usage in (MB/sec). The primary focus is to understand the usage of cloud memory due to the increase in the number of requests and policy entities. For 10 and 100 policy entities and 10 and 100 requests, the results show that the memory usage is low compared to other greater sets of requests and policies. For 1000 policy entities and 1000 requests at a time, the memory usage increases by 3.74% and 4.45% respectively for 10 and 100 requests while the number of policy entities is same. For higher (10000) policy entities and 10000 requests, the memory usage increases much higher than other sets of requests.

### 5.3.5 System Response Time:

The system response time is the total round trip time (RTT) of the whole architecture. In the fourth experiment, the mobile client sends a POST request to the cloud, cloud verifies it and forwards the message to the gateway server which has a running HTTP server. The HTTP server receives the message and converts it to the CoAP message and sends to the CoAP client. The CoAP clients send the response back, and it goes all the way to the mobile client through the cloud. The total interaction time is estimated as system response time. The whole procedure is completed with 10 POST requests with varying policy entity sets of 10, 100, 1000 and 10000. The result is later compared with an existing access control system in IoT [5] as shown in figure 5.15. The existing architecture does not vary due to any policy entities, and that is why the value is always constant.

Figure 5.15 also shows that the proposed architecture is having a better response time than the existing system [5] in each case expect when the policy entities reach to 10000 because it takes much time to find the optimal matching rule from a large number of policy entities. The overall system comparison shows that the proposed architecture does not introduce any delay due to the introduction of cloud services.

### 5.3.6 System Response Time for Different Request Type:

In this experiment, the system response time is of the whole architecture is measured for GET, POST, and PUT requests. The mobile client sends a request to the cloud and based on the request type the cloud either forwards it to the gateway server or instantly replies back to the mobile client. The whole procedure is

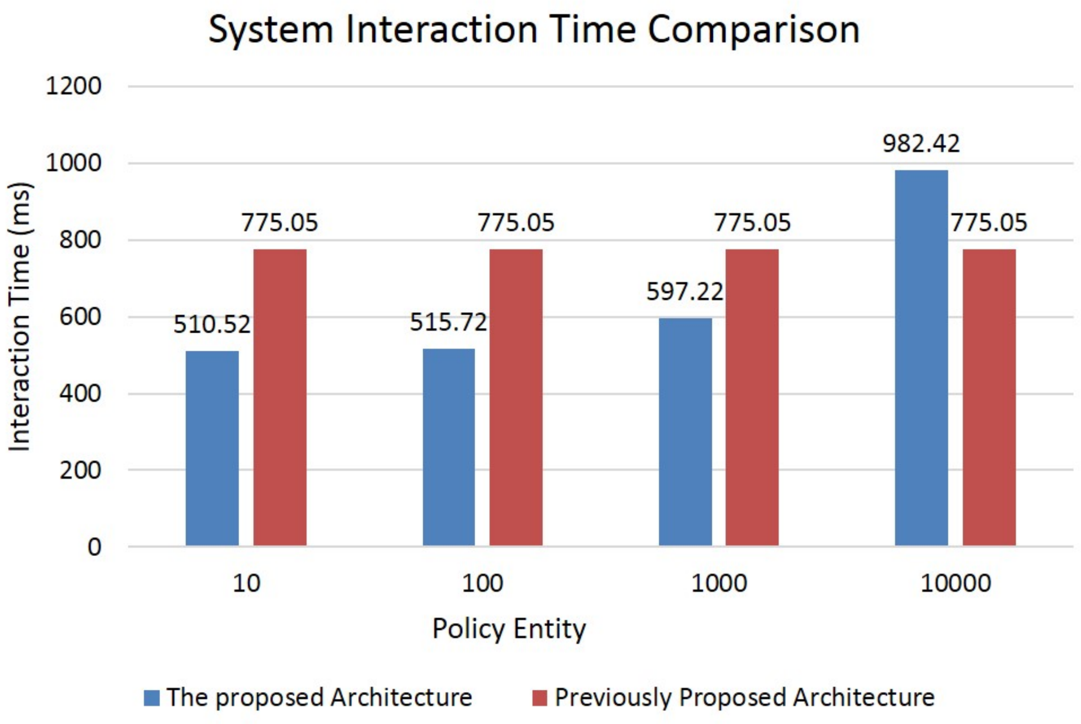


**Figure 5.15:** Calculate Overall System Response Time. The red column represents the overall system interaction time in [5]

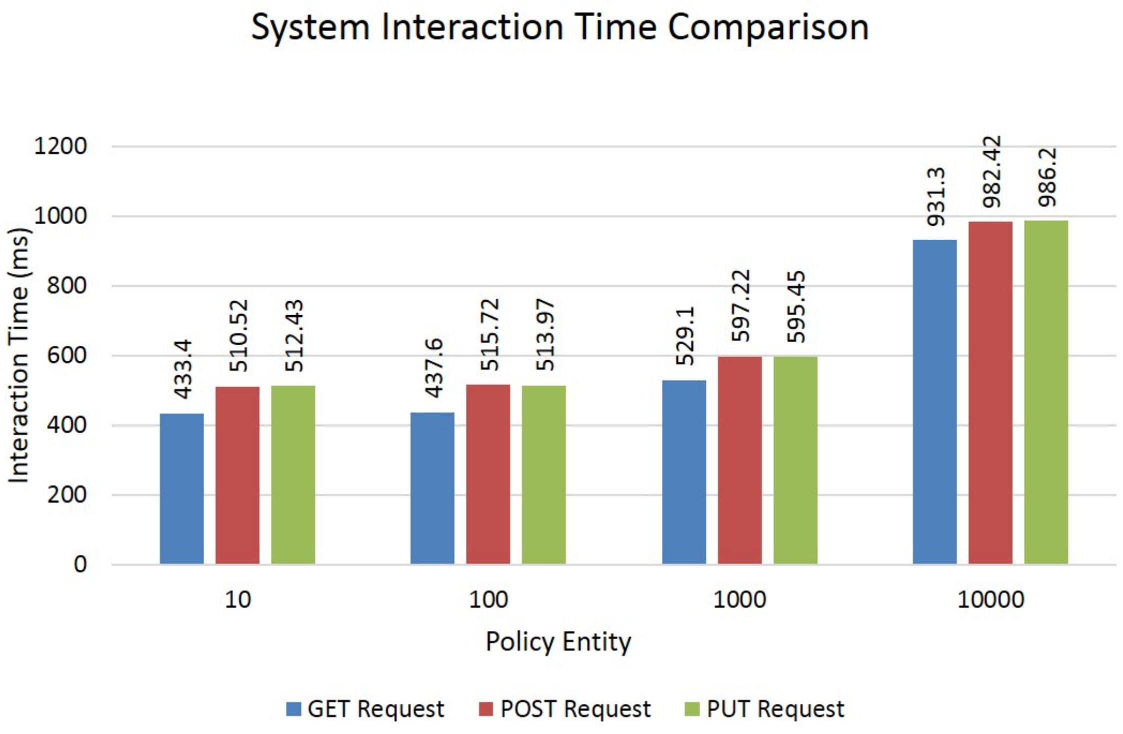


**Figure 5.16:** Calculate Overall System Response Time for GET, POST, and PUT Request.

completed with 10 requests with varying policy entity sets of 10, 100, 1000 and 10000. The interaction time is compared with each other to analyze the time required to complete each type of requests.

Figure 5.16 shows that the GET request requires less time than other two as the cloud can reply back to the mobile client as soon as the authorization is completed. The POST and PUT request go all the way to the sensors, so these requests take a longer time to complete the whole task. However, these requests still take less time than the existing system [5] unless the number of policy entities reaches to 10000.

### 5.3.7 Scalability

The cloud plays a vital role managing the interaction between mobile and IoT resources. Each mobile client differs from others by their device IDs. Each IoT resource also varies from each other, and CoAP server connects to the cloud. So the cloud needs to handle a good number of requests to maintain the availability of the service. In recent years, the cloud has become a powerful machine and capable of handling large datasets and IO requests. In this thesis, Google App Engine is used to host cloud application. Google App Engine is capable of automatic scaling which is more than enough to support a home automation [161]. Google App Engine has no restrictions on incoming bandwidths (approx. 660,000 calls/minute). Also, the size of a single request and response can be up to 32 MB, which can reduce the number of round trips from the client application, which is especially useful for environments, such as mobile devices with high latency [162]. Google App Engine has the option to create multiple entities to divide the workload to support the large industry or corporate. The support of scalability is one of the main reason to choose cloud to host the access control application.

## 5.4 Discussion

This Chapter described the various experiments that were conducted to evaluate our system's solutions, the challenges of interaction time of each component and the overall system, CPU usage, and memory usage of asynchronous and synchronize requests. The first part of the evaluation focused on the measurement of interaction time due to the addition of a cloud service. The communication time employed simulation in a laboratory to determine how long it takes for data to be accessed on sensors. The requester is the mobile client, middleware is the cloud, and the provider is the gateway server. The evaluation focused mainly on requester-middleware, middleware-provider, and requester-middleware-provider interactions. Besides, since the performance of systems slows down under heavy workloads, resulting in high latency, scalability analysis of cloud was also conducted.

The summary of the experiments is listed below based on the goals. Although different sets of policy entity are used, the discussion is focused on the results of 1000 policy entity.

### 5.4.1 Testing Interaction Time to Access Sensor's Data

**Mobile-Cloud Interaction Time:** Evaluated interaction time between an Android mobile application (mobile client/requester) and Google App Engine (middleware). Although the proposed system adds additional verification in the cloud, the result shows that the proposed system does not take much time than the Username-Password authentication system for policy entity 10, 100 and 1000. The result also shows that if multiple requests are sent together after a random delay (asynchronously), the interaction time difference between different policy entities(100,1000) do not vary greater than 13%.

**CoAP-Cloud Interaction Time:** Evaluated interaction time between the Google App Engine (middleware) and sensor devices (providers). The CoAP system includes cloud interaction, and that is why it takes more time to complete the task than the existing solution [5].

### 5.4.2 Testing CPU Usage to Access Sensor's Data

**Requester-Middleware CPU Usage:** Evaluated CPU Usage of middleware (Google App Engine) when the requests are being processed. The requests in this experiment are sent together after a random interval, and the CPU usage of cloud is measured. For 1000, the result shows that the cloud uses 143 cycles in a second when the policy entity increases from 100 to 1000.

**Middleware-Provider CPU Usage:** Evaluated CPU Usage of Gateway servers (Raspberry PI, Intel Edison, MACBOOK Pro). The requests are sent in two categories: Synchronize and Asynchronous. For both 1000 requests, Intel Edison uses more CPU to complete the tasks. It is also visible that, Raspberry PI uses less CPU when the requests are made asynchronously.

### 5.4.3 Testing Memory Usage to Access Sensor's Data

**Requester-Middleware Memory Usage:** Evaluated memory Usage of middleware (Google App Engine) when the requests are being processed. The requests in this experiment are sent together after a random interval, and the memory usage of cloud is measured. For 1000 requests, the result shows that the cloud uses 4 MB in a second when the policy entity increases from 100 to 1000.

**Middleware-Provider Memory Usage:** Evaluated Memory Usage of Gateway servers (Raspberry PI, Intel Edison, MACBOOK Pro). The requests are sent in two categories: Synchronize and Asynchronous. For both 1000 requests, Intel Edison takes more memory to complete the tasks (approx 8151 KB) whereas Raspberry PI uses 6636 KB memory for 1000 requests. It is also visible that, Raspberry PI uses similar memory irrespective of the request processing types.

### 5.4.4 Testing System Interaction Time to Access Sensor's Data

Evaluated total system interaction time and compared with the existing solution [5]. For 1000 policy entity, the proposed solution takes approx 22% less time than the existing solution [5].

### 5.4.5 Testing System Interaction Time to Access Sensor's Data for Different Request Types

Evaluated total system interaction time for different request types. GET requests always take less time because these kinds of requests do not need to go to the gateway servers or sensors. The cloud delivers the value directly to the requester. For POST and PUT request types, the interaction time is similar to both types of requests to go to the gateway server to insert or update a value.

## 5.5 Summary

This chapter focuses on various experiments that were conducted to evaluate the proposed architecture for factors such as interaction time and scalability of the system. Section 5.1 describes the implementation details and experimental setup for conducting proposed experiments. Section 5.2 describes the performance matrices for the experiments. Section 5.3 includes the list of experiments that are conducted to test the proposed architecture. Section 5.4 discusses about the experimental results. The proposed architecture is expected to answer our research question and minimize the threats on the sensors.

# Chapter 6
# Conclusion and Future Work

## 6.1 Summary

This research has shown that dynamic access control system can implement in IoT environment. Using REST as architectural design, IoT services can be efficiently built and deployed to control an IoT environment. However, there are challenges such as trust, authentication, delay, and system overhead due to incorporating mobile devices, cloud, and sensors. Sensors are a low-powered device with memory constraints, and that is why it is hard to implement existing network security solutions in the sensors. The focus of this work is to ensure proper authorization system for IoT by introducing Cloud and attribute-based access control.

This research proposes an architecture that introduced a dynamic access control system that can give a user access to the IoT resources considering the context information from the user's mobile device (as shown in Figure 6.1). The proposed method approves an access for a session, and the resources can be obtained based on the permission level implied by the System Administrator. The contribution of this work is to ensure the sufficient security of the IoT resources as well as keeping the user experience at a high level by allowing the device for personal and professional use.

Initially, this architecture examines the authorization requests at the device level and later in the cloud. Adding two layers of security to access IoT's resources make the approach more reliable and without adding delay. Also in future, keeping the user's request history will give the user a chance to get the access even if the request does not match well enough with the organization's policies.

Existing solutions in IoT access controls are analyzed before improvements are offered. The protocol was implemented and evaluated on Raspberry PI 3, Google App Engine and Android mobile devices. These hardware configurations are commonly used in IoT applications. Evaluation results show that for a standard number of rules, the interaction time of this proposed architecture took 23% less time to complete the whole interaction. Although cloud is included, the result shows that it does not introduce any latency. The CPU usage and memory usage is also at a minimal level. Moreover, this approach becomes useful when storing and analyzing the sensor data for further researches.

The solution offers feasible, fast, and reliable IoT environment. This approach is recommended where the application requires dynamic centralize access, storing and computing power and engaging portable devices. It is possible to replace Google Cloud with any other options as the cloud application is independent in the

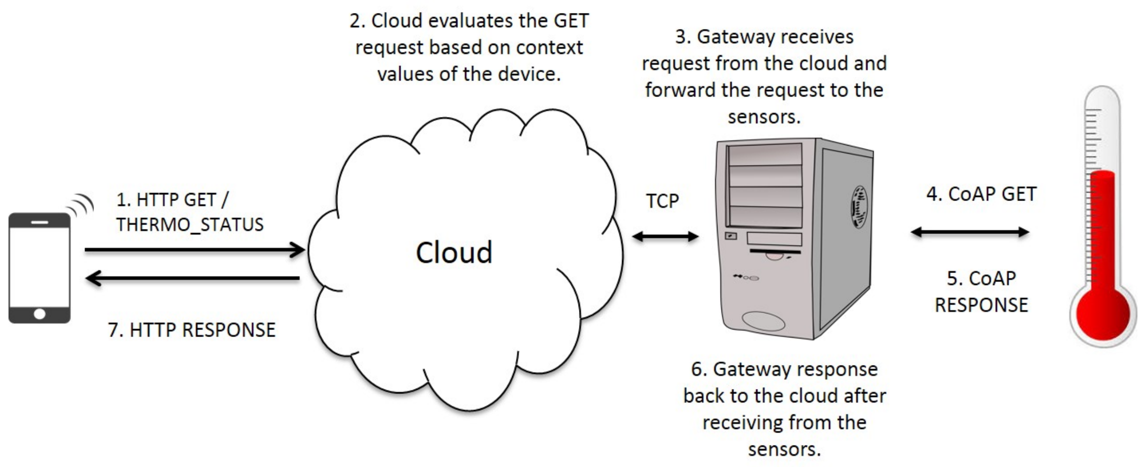

2. Cloud evaluates the GET request based on context values of the device.
3. Gateway receives request from the cloud and forward the request to the sensors.
1. HTTP GET / THERMO_STATUS
Cloud
TCP
4. CoAP GET
5. CoAP RESPONSE
7. HTTP RESPONSE
6. Gateway response back to the cloud after receiving from the sensors.

**Figure 6.1:** The proposed solution of the scenario.

scenario. Also, any other protocol can be used in sensor level as long as it offers same security level as CoAP does.

The contribution of this research work are summarized below:

- It is feasible to implement an attribute-based dynamic access control system for IoT environment.
- The mobile context values are used to ensure the validity of user in IoT environment.
- The architecture completes a request with less interaction time and by using less CPU and memory usage.
- The use of CoAP protocol enhances the possibility of using any low-powered devices as a gateway server.
- The use of REST framework ensures the data security between the mobile device and the cloud.
- The configuration file is easy to maintain and does not require to compile in order to update.

## 6.2 Future Works

The primary focus was the scenario when one mobile client interacts with a sensor device. In this thesis, the implementation was done to provide a proof-of-concept. The interaction time, CPU usage, and memory usage was measured to proof that the system is fast and reliable. The interaction time and the system overhead were only measured. The future plans mostly involve designing more experiments with real-life malicious data and integrating user behavioral pattern.

- **Introduction of Behavior:** By storing users requests and applying machine learning algorithm, the system can be extended to evaluate the requests against user behavioral pattern as shown in Figure 6.2.

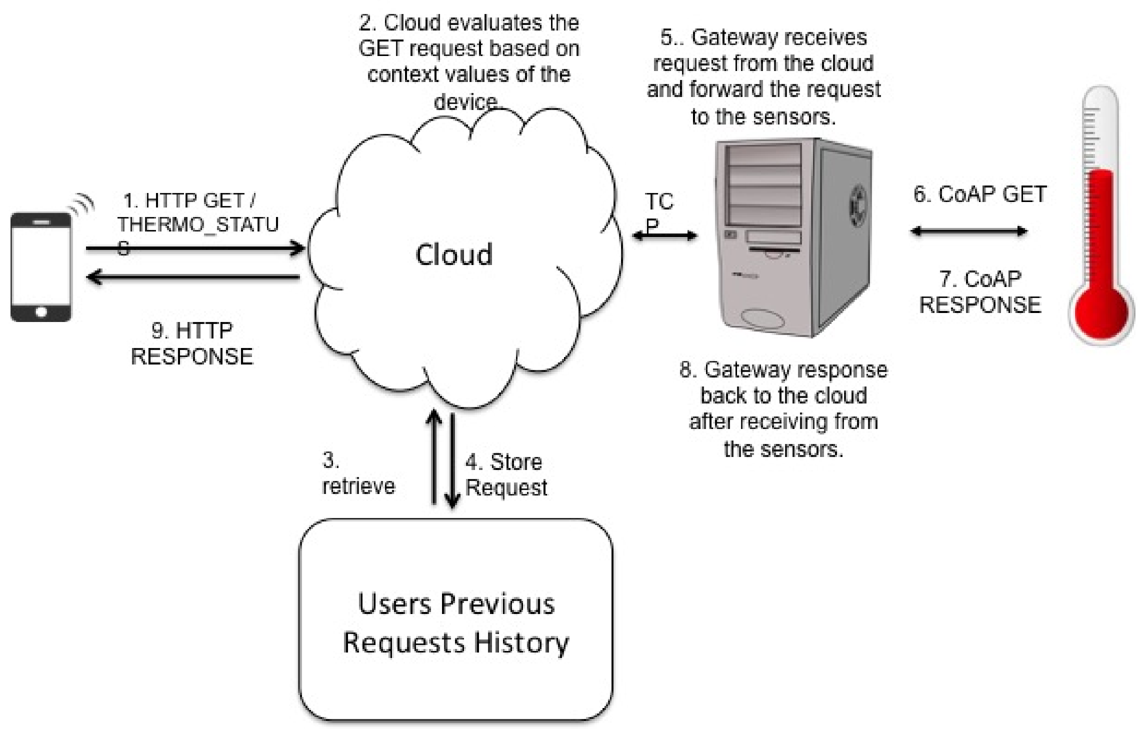

2. Cloud evaluates the GET request based on context values of the device.
5.. Gateway receives request from the cloud and forward the request to the sensors.
1. HTTP GET / THERMO_STATUS
TCP
6. CoAP GET
Cloud
7. CoAP RESPONSE
9. HTTP RESPONSE
8. Gateway response back to the cloud after receiving from the sensors.
3. retrieve
4. Store Request
Users Previous Requests History

**Figure 6.2:** The Extended Architecture.

The requests will be stored and later pattern matching or machine learning algorithm can be used to identify a user.

- **Weighted Context Values:** Further experiments can be designed to understand the important context values by weighting them individually. By varying the weights, we can have a better understanding of which context values are needed and which are redundant.
- **Real life Malware:** The experiments do not provide substantial proof of security. It will be interesting to see how the system behaves if we send malicious data to the cloud. These experiments will help us to understand how well the proposed architecture works regarding security.
- **Different types of Access Control:** Experiments can be designed to compare between ABAC, RBAC, and different other access control systems in terms of security and reliability.
- **Compare Between Trust Models:** There are a lot of existing trust models available right now, and these trust models can be implemented in this system to evaluate which model performs best.

# Bibliography

[1] Preston Walters and Samir Vyas. Managing IoT Data from the Edge to Cloud: Your assets are talking - ARE YOU LISTENING? In *InterConnect 2016 The Premier Cloud & Mobile Conference*, pages 1–27, Las Vegas, 2016.

[2] Ning YE, Yan Zhu, Ru-chuan WANG, Reza Malekian, and Lin Qiao-min. An Efficient Authentication and Access Control Scheme for Perception Layer of Internet of Things. *International Journal of Applied Mathematics & Information Sciences*, 8(4):1617–1624, 2014.

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

[4] Yuanjun Song. *Security in Internet of Things*. PhD thesis, Royal Institute of Technology, 2013.

[5] Denis Sitenkov. *Access Control in the Internet of Things*. PhD thesis.

[6] Improvements in Cell Phone Technology — eHow. `http://www.ehow.com/about{_}5542192{_}improvements-cell-phone-technology.html`.

[7] José Luis Hernández-Ramos, Antonio J. Jara, Leandro Marín, and Antonio F. Skarmeta. Distributed Capability-Based Access Control for the Internet of Things. In *5th International Workshop on Managing Insider Security Threats (MIST 2013)*, oct 2013.

[8] Rahul Godha and Sneh Prateek. Home Automation: Access Control for IoT Devices. *International Journal of Scientific and Research Publications (IJSRP)*, 4(10), 2014.

[9] Jing Liu, Yang Xiao, and C.L. Philip Chen. Internet of things' authentication and access control. *International Journal of Security and Networks*, 7(4):228–241, apr 2012.

[10] Lu Yan, Yan Zhang, Laurence T. Yang, and Huansheng Ning. *The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems*. Auerbach Publications, Boston, mar 2008.

[11] Lu Tan and Neng Wang. Future internet: The Internet of Things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 5, pages 376–380. IEEE, aug 2010.

[12] Alessandro Bassi. *Enabling things to talk: Designing IoT solutions with the IoT architectural reference model*. 2013.

[13] Yun-kyung Lee, Jae-deok Lim, Yong-seong Jeon, and Jeong-nyeo Kim. Technology trends of access control in IoT and requirements analysis. In *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1031–1033. IEEE, oct 2015.

[14] S. Sicari, A. Rizzardi, L.A. Grieco, and A. Coen-Porisini. Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76:146–164, nov 2014.

[15] Pablo Punal Pereira, Jens Eliasson, and Jerker Delsing. An authentication and access control framework for CoAP-based Internet of Things. In *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, pages 5293–5299. IEEE, oct 2014.

[16] Thomas Kimsey, Jason Jeffords, Yassi Moghaddam, and Andrzej Rucinski. An IoT Based Service System as a Research and Educational Platform. In Dariusz Barbucha, Ngoc Thanh Nguyen, and John Batubara, editors, *New Trends in Intelligent Information and Database Systems*, volume 598 of *Studies in Computational Intelligence*, pages 249–257. Springer International Publishing, Cham, 2015.

[17] Priyanka Upadhyay, Gurpreet Matharu, and Naveen Garg. Modeling Agility in Internet of Things (IoT) Architecture. In J. K. Mandal, Suresh Chandra Satapathy, Manas Kumar Sanyal, Partha Pratim Sarkar, and Anirban Mukhopadhyay, editors, *Information Systems Design and Intelligent Applications*, volume 340 of *Advances in Intelligent Systems and Computing*, pages 779–786. Springer India, New Delhi, 2015.

[18] Mark O'Neill. The Internet of Things: do more devices mean more risks? *Computer Fraud & Security*, 2014(1):16–17, jan 2014.

[19] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, oct 2010.

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

[21] Maria Rita Palattella, Nicola Accettura, Xavier Vilajosana, Thomas Watteyne, Luigi Alfredo Grieco, Gennaro Boggia, and Mischa Dohler. Standardized Protocol Stack for the Internet of (Important) Things. *IEEE Communications Surveys & Tutorials*, 15(3):1389–1406, jan 2013.

[22] Olivier Hersent, David Boswarthick, and Omar Elloumi. *The Internet of Things: Key Applications and Protocols.* John Wiley and Sons Ltd., 2012.

[23] Dave Evans. The Internet of Things How the Next Evolution of the Internet Is Changing Everything. Technical report, Cisco Internet Business Solutions Group (IBSG), 2011.

[24] Bruce Ndibanje, Hoon-Jae Lee, and Sang-Gon Lee. Security analysis and improvements of authentication and access control in the Internet of Things. *Sensors (Basel, Switzerland)*, 14(8):14786–14805, jan 2014.

[25] Huansheng Ning and Sha Hu. Technology classification, industry, and education for Future Internet of Things. *International Journal of Communication Systems*, 25(9):1230–1241, sep 2012.

[26] Ebraheim Alsaadi and Abdallah Tubaishat. Internet of Things: Features, Challenges, and Vulnerabilities. *International Journal of Advanced Computer Science and Information Technology (IJACSIT)*, 4(1):1–13, 2015.

[27] Salvatore Gaglio and Giuseppe Lo Re. *Advances onto the Internet of Things: How Ontologies Make the Internet of Things Meaningful*. Springer International Publishing, 1 edition, 2014.

[28] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi. From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view. *IEEE Wireless Communications*, 17(6):44–51, dec 2010.

[29] Rolf H. Weber. Internet of Things  New security and privacy challenges. *Computer Law & Security Review*, 26(1):23–30, jan 2010.

[30] Rolf H. Weber and Romana Weber. *Internet of Things: Legal Perspectives*. Springer-Verlag Berlin Heidelberg, 1 edition, 2010.

[31] Bruce D. Weinberg, George R. Milne, Yana G. Andonova, and Fatima M. Hajjat. Internet of Things: Convenience vs. privacy and secrecy. *Business Horizons*, 58(6):615–624, aug 2015.

[32] Leading Tools Manufacturer Transforms Operations with IoT. Technical report, 2014. `http://www.cisco.com/c/dam/en{_}us/solutions/industries/docs/manufacturing/c36-732293-00-stanley-cs.pdf`.

[33] Rolf H. Weber. Internet of things: Privacy issues revisited. *Computer Law & Security Review*, 31(5):618–627, aug 2015.

[34] Tony Bradley. Experts pick the top 5 security threats for 2015, 2015. `http://www.pcworld.com/article/2867566/experts-pick-the-top-5-security-threats-for-2015.html`.

[35] Tejaswini Herath and H Raghav Rao. Protection motivation and deterrence: a framework for security policy compliance in organisations. *European Journal of Information Systems*, 18(2):106–125, apr 2009.

[36] Michael S. Smith. Protecting Privacy in an IoT-Connected World. *Information Management*, 49(6), nov 2015.

[37] Robin Kravets, Güliz Seray Tuncay, and Hari Sundaram. For Your Eyes Only. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services - MCS '15*, pages 28–35, New York, New York, USA, sep 2015. ACM Press.

[38] Joseph Turow. *The Daily You: How the New Advertising Industry Is Defining Your Identity and Your Worth*. Yale University Press, 2011.

[39] The Big Data Security Gap: Protecting the Hadoop Cluster. Technical report, 2013. `http://www.zettaset.com/wp-content/uploads/2014/04/zettaset{_}wp{_}security{_}0413.pdf`.

[40] Ismail Butun, Burak Kantarci, and Melike Erol-Kantarci. Anomaly detection and privacy preservation in cloud-centric Internet of Things. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 2610–2615. IEEE, jun 2015.

[41] Jeff Holleran. Building a Better BYOD Strategy. *Risk Management*, 61(7):12–13, 2014.

[42] Yong Wang, Kevin Streff, and Sonell Raman. Smartphone Security Challenges. *Computer*, 45(12):52–58, dec 2012.

[43] Yong Wang, Jinpeng Wei, and K. Vangury. Bring your own device security issues and challenges. In *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, pages 80–85. IEEE, jan 2014.

[44] 2011 Mobile Threats Report. Technical report, 2012.

[45] Alastair Stevenson. Android malware makes up 79 percent of total threats, warns US Department of Homeland Security, 2013. `http://www.v3.co.uk/v3-uk/news/2291561/android-malware-makes-up-79-percent-of-total-threats-warns-us-department-of-homeland-security`.

[46] Hormazd Romer. Best practices for BYOD security. *Computer Fraud & Security*, 2014(1):13–15, jan 2014.

[47] Antony Adshead. Data set to grow 10-fold by 2020 as internet of things takes off., 2014. `http://www.computerweekly.com/news/2240217788/Data-set-to-grow-10-fold-by-2020-as-internet-of-things-takes-off`.

[48] Data Distribution Service for Real-time Systems, V 1.2. Technical report, Object Management Group, 2007. `http://www.omg.org/spec/DDS/1.2/`.

[49] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347 – 2376, 2015.

[50] Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (CoAP)., 2014. `https://tools.ietf.org/html/rfc7252`.

[51] Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan, and Colin Keng-Yan Tan. Performance evaluation of MQTT and CoAP via a common middleware. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6. IEEE, apr 2014.

[52] Niccolo De Caro, Walter Colitti, Kris Steenhaut, Giuseppe Mangino, and Gianluca Reali. Comparison of two lightweight protocols for smartphone-based sensing. In *2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*, pages 1–6. IEEE, nov 2013.

[53] Leonard Richardson, Mike Amundsen, and Sam Ruby. CoAP: REST for Embedded Systems. In *RESTful Web APIs*, pages 287–293. O'Reilly Media, 2013.

[54] Carsten Bormann. CoAP, 2014. `http://coap.technology/`.

[55] Ilpo Suominen. ACCESS CONTROL FOR INTERNET OF THINGS, 2015. `https://www.intopalo.com/access-control-for-internet-of-things`.

[56] Jaehong Park and Ravi Sandhu. Towards usage control models: beyond traditional access control. In *Proceedings of the seventh ACM symposium on Access control models and technologies - SACMAT '02*, pages 57–64, New York, New York, USA, jun 2002. ACM Press.

[57] Guoping Zhang and Wentao Gong. The Research of Access Control Based on UCON in the Internet of Things. *Journal of Software*, 6(4):724–731, apr 2011.

[58] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. A capability-based security approach to manage access control in the Internet of Things. *Mathematical and Computer Modelling*, 58(5-6):1189–1205, sep 2013.

[59] Rui Zhang, Yanchao Zhang, and Kui Ren. Distributed Privacy-Preserving Access Control in Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(8):1427–1438, aug 2012.

[60] Guoping Zhang and Jiazheng Tian. An extended role based access control model for the Internet of Things. In *2010 International Conference on Information, Networking and Automation (ICINA)*, volume 1, pages V1–319–V1–323. IEEE, oct 2010.

[61] ADAPTIVE ACCESS FRAMEWORK: CONCEPT, DEFINITION, APPLICATION. Technical report, ISSA End-to-End Trust Working Group, 2012.

[62] Yan Ling Zhao. Research on Data Security Technology in Internet of Things. *Applied Mechanics and Materials*, 433-435:1752–1755, oct 2013.

[63] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brünig, and Georg Carle. DTLS based security and two-way authentication for the Internet of Things. *Ad Hoc Networks*, 11(8):2710–2723, nov 2013.

[64] Rodrigo Roman, Cristina Alcaraz, Javier Lopez, and Nicolas Sklavos. Key management systems for sensor networks in the context of the Internet of Things. *Computers & Electrical Engineering*, 37(2):147–159, mar 2011.

[65] Wenliang Du, Jing Deng, Yunghsiang S. Han, Pramod K. Varshney, Jonathan Katz, and Aram Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security*, 8(2):228–258, may 2005.

[66] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communication security - CCS '03*, pages 52–61, New York, New York, USA, oct 2003. ACM Press.

[67] Ilung Pranata, Rukshan I. Athauda, and Geoff Skinner. Securing and Governing Access in Ad-Hoc Networks of Internet of Things. In *Engineering and Applied Science*, pages 84–90, Calgary,AB,Canada, dec 2012. ACTAPRESS.

[68] Hong Ning. A Security Framework for the Internet of Things Based on Public Key Infrastructure. *Advanced Materials Research*, 671-674:3223–3226, mar 2013.

[69] Zhen-Qiang WU, Yan-Wei ZHOU, and Jian-Feng MA. A Security Transmission Model for Internet of Things. *Chinese Journal of Computers*, 34(8):1351–1364, oct 2011.

[70] Jun-Ya Lee, Wei-Cheng Lin, and Yu-Hung Huang. A lightweight authentication protocol for Internet of Things. In *2014 International Symposium on Next-Generation Electronics (ISNE)*, pages 1–2. IEEE, may 2014.

[71] Haodong Wang, Bo Sheng, and Qun Li. Elliptic curve cryptography-based access control in sensor networks. *International Journal of Security and Networks*, 1(3/4):127, dec 2006.

[72] Hsiu-Lien Yeh, Tien-Ho Chen, Pin-Chuan Liu, Tai-Hoo Kim, and Hsin-Wen Wei. A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors (Basel, Switzerland)*, 11(5):4767–4779, jan 2011.

[73] B. B. P. Rao, P. Saluia, N. Sharma, A. Mittal, and S. V. Sharma. Cloud computing for Internet of Things & sensing based applications. In *2012 Sixth International Conference on Sensing Technology (ICST)*, pages 374–380. IEEE, dec 2012.

[74] The Internet of Things: Hype and Potential, 2014. `http://tech.globant.com/en/demystifying-the-internet-of-things/`.

[75] Ann Cavoukian, Michelle Chibba, Graham Williamson, and Andrew Ferguson. The Importance of ABAC: Attribute-Based Access Control to Big Data: Privacy and Context. Technical report, The Privacy and big Data Institute, 2015.

[76] A. Boukerch, L. Xu, and K. EL-Khatib. Trust-based security for wireless ad hoc and sensor networks. *Computer Communications*, 30(11):2413–2427, 2007.

[77] Haiguang Chen, Huafeng Wu, Xi Zhou, and Chuanshan Gao. Agent-based Trust Model in Wireless Sensor Networks. In *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, volume 3, pages 119–124. IEEE, jul 2007.

[78] Avinash Srinivasan, Joshua Teitelbaum, and Jie Wu. DRBTS: Distributed Reputation-based Beacon Trust System. In *2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 277–283. IEEE, 2006.

[79] Félix Gómez Mármol and Gregorio Martínez Pérez. Providing trust in wireless sensor networks using a bio-inspired technique. *Telecommunication Systems*, 46(2):163–180, feb 2011.

[80] Dong Chen, Guiran Chang, Dawei Sun, Jiajia Li, Jie Jia, and Xingwei Wang. TRM-IoT: A Trust Management Model Based on Fuzzy Reputation for Internet of Things. *Computer Science and Information Systems*, 8(4):1207–1228, 2011.

[81] Fenye Bao and Ing-Ray Chen. Dynamic trust management for internet of things applications. In *Proceedings of the 2012 international workshop on Self-aware internet of things - Self-IoT '12*, pages 1–6, New York, New York, USA, 2012. ACM Press.

[82] Fenye Fenye Bao and Ing-Ray Ing-Ray Chen. Trust management for the internet of things and its application to service composition. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6. IEEE, jun 2012.

[83] Michele Nitti, Roberto Girau, Luigi Atzori, Antonio Iera, and Giacomo Morabito. A subjective model for trustworthiness evaluation in the social Internet of Things. In *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, pages 18–23. IEEE, sep 2012.

[84] Raquel Lacuesta, Guillermo Palacios-Navarro, Carlos Cetina, Lourdes Pe{\~{n}}alver, and Jaime Lloret. Internet of things: where to be is to trust. *EURASIP Journal on Wireless Communications and Networking*, 2012(1):1–16, 2012.

[85] Parikshit N. Mahalle, Pravin A. Thakre, Neeli Rashmi Prasad, and Ramjee Prasad. A fuzzy approach to trust based access control in internet of things. In *Wireless VITAE 2013*, pages 1–5, Atlantic City, jun 2013. IEEE.

[86] Jing Pei Wang, Sun Bin, Yang Yu, and Xin Xin Niu. Distributed Trust Management Mechanism for the Internet of Things. *Applied Mechanics and Materials*, 347-350:2463–2467, aug 2013.

[87] Wen-Mao LIU, Li-Hua YIN, Bin-Xing FANG, and Hong-Li ZHANG. A Hierarchical Trust Model for the Internet of Things. *Chinese Journal of Computers*, 35(5):846–855, nov 2012.

[88] Yosra Ben Saied, Alexis Olivereau, Djamal Zeghlache, and Maryline Laurent. Trust management system design for the Internet of Things: A context-aware and multi-service approach. *Computers & Security*, 39:351–365, 2013.

[89] P. Dong, J. F. Guan, X. P. Xue, and H. C. Wang. Attack-Resistant Trust Management Model Based on Beta Function for Distributed Routing in Internet of Things. *Wireless Communication over ZigBee for Automotive Inclination Measurement. China Communications*, 9(4):89–98, 2012.

[90] Tao Liu, Ya Wen Guan, Yi Qun Yan, Li Liu, and Qi Chao Deng. A WSN-Oriented Key Agreement Protocol in Internet of Things. *Applied Mechanics and Materials*, 401-403:1792–1795, sep 2013.

[91] Insights on the Current State of BYOD. Technical report, Intel IT Center, 2012.

[92] Vijay Sivaraman, Hassan Habibi Gharakheili, Arun Vishwanath, Roksana Boreli, and Olivier Mehani. Network-level security and privacy control for smart-home IoT devices. In *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 163–167. IEEE, oct 2015.

[93] Reijo M. Savola, Pekka Savolainen, Antti Evesti, Habtamu Abie, and Markus Sihvonen. Risk-driven security metrics development for an e-health IoT application. In *2015 Information Security for South Africa (ISSA)*, pages 1–6. IEEE, aug 2015.

[94] Rakesh Kumar and Hardeep Singh. A Proactive Procedure to Mitigate the BYOD Risks on the Security of an Information System. *ACM SIGSOFT Software Engineering Notes*, 40(1):1–4, feb 2015.

[95] Dennis Titze, P. Stephanow, and J. Schutte. A Configurable and Extensible Security Service Architecture for Smartphones. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 1056–1062. IEEE, mar 2013.

[96] Sean Chung, Sam Chung, Teresa Escrig, Yan Bai, and Barbara Endicott-Popovsky. 2TAC: Distributed Access Control Architecture for "Bring Your Own Device" Security. In *2012 ASE/IEEE International Conference on BioMedical Computing (BioMedCom)*, pages 123–126. IEEE, dec 2012.

[97] Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. Trusted execution environments on mobile devices. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*, pages 1497–1498, New York, New York, USA, nov 2013. ACM Press.

[98] Zhibo Zhao and Fernando C. Colon Osono. TrustDroid: Preventing the use of SmartPhones for information leaking in corporate networks through the used of static analysis taint tracking. In *2012 7th International Conference on Malicious and Unwanted Software*, pages 135–143. IEEE, oct 2012.

[99] Jason H. Christensen. Using RESTful web-services and cloud computing to create next generation mobile applications. In *Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pages 627–634, New York, New York, USA, oct 2009. ACM Press.

[100] Will Shanklin. 2015 Smartphone Comparison Guide, 2015. `http://www.gizmag.com/2015-smartphone-comparison-guide-1/36992/`.

[101] Friedemann Mattern and Christian Floerkemeier. From Active Data Management to Event-Based Systems and More. In Kai Sachs, Ilia Petrov, and Pablo Guerrero, editors, *From the Internet of Computers to the Internet of Things*, volume 6462 of *Lecture Notes in Computer Science*, pages 242–259. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[102] Michael Hausenblas. Smart Phones and the Internet of Things, 2014. `https://www.mapr.com/blog/smart-phones-and-internet-things`.

[103] NOKIA. A Holistic Approach to Business Mobility. Technical report, London, 2006. `http://dailywireless.tradepub.com/free-offer/a-holistic-approach-to-business-mobility/w{_}aaaa993?sr=hicat{&}{_}t=hicat:759`.

[104] Sheikh Mahbub Habib, Sebastian Ries, and Max Muhlhauser. Cloud Computing Landscape and Research Challenges Regarding Trust and Reputation. In *2010 7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing*, pages 410–415. IEEE, oct 2010.

[105] Stephen Marsh, Pamela Briggs, Khalil El-Khatib, Babak Esfandiari, and John A. Stewart. Defining and Investigating Device Comfort. *Information and Media Technologies*, 6(3):914–935, sep 2011.

[106] Mark A. Harris and Karen P. Patten. Mobile device security considerations for small- and medium-sized enterprise business mobility. *Information Management & Computer Security*, 22(1):97–114, mar 2014.

[107] Number of apps available in leading app stores as of July 2015, 2015. `http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/`.

[108] Number of mobile app downloads worldwide from 2009 to 2017 (in millions), 2015. `http://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/`.

[109] Sascha Wessel, Manuel Huber, Frederic Stumpf, and Claudia Eckert. Improving mobile device security with operating system-level virtualization. *Computers & Security*, 52:207 – 220, apr 2015.

[110] Yury Namestnikov and Denis Maslennikov. Kaspersky Security Bulletin 2012., 2012. `https://securelist.com/analysis/kaspersky-security-bulletin/36703/kaspersky-security-bulletin-2012-the-overall-statistics-for-2012/`.

[111] Ellen Messmer. Android malware exploding, says Trend Micro, 2012. http://www.networkworld.com/article/2160862/network-security/android-malware-exploding--says-trend-micro.html.

[112] THREAT REPORT H2 2014. Technical report, 2014. https://www.f-secure.com/documents/996508/1030743/Threat{_}Report{_}H2{_}2014.

[113] 2014 MOBILE THREAT REPORT. Technical report, 2014. https://www.lookout.com/img/images/Consumer{_}Threat{_}Report{_}Final{_}ENGLISH{_}1.14.pdf.

[114] D. B. Audretsch and A. Roy Thurik. What's New about the New Economy? Sources of Growth in the Managed and Entrepreneurial Economies. *Industrial and Corporate Change*, 10(1):267–315, mar 2001.

[115] Denis Maslennikov. Find and Call: Leak and Spam, 2012. https://securelist.com/blog/incidents/33544/find-and-call-leak-and-spam-57/.

[116] State of Mobile Security 2012. Technical report, 2012. https://www.lookout.com/img/images/lookout-state-of-mobile-security-2012.pdf.

[117] Alexios Mylonas, Anastasia Kastania, and Dimitris Gritzalis. Delegate the smartphone user? Security awareness in smartphone platforms. *Computers & Security*, 34:47–66, may 2013.

[118] Cisco 2014 Annual Security Report. Technical report, CISCO, 2014.

[119] Derek Halliday. 2013 Mobile Threat Predictions, 2012. https://blog.lookout.com/blog/2012/12/13/2013-mobile-threat-predictions/.

[120] Mark A. Harris, Steven Furnell, and Karen Patten. Comparing the Mobile Device Security Behavior of College Students and Information Technology Professionals. *Journal of Information Privacy and Security*, 10(4):186–202, jan 2015.

[121] State of Security in the App Economy: Mobile Apps Under Attack. Technical report, ARXAN, 2012. https://www.arxan.com/wp-content/uploads/assets1/pdf/state-of-security-app-economy.pdf.

[122] Yury Gadyatskaya, Olga Massacci, Fabio Zhauniarovich. Security in the Firefox OS and Tizen Mobile Platforms. *IEEE Computer*, 47(6):57–63, 2014.

[123] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a Better Understanding of Context and Context- Awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, 1999. Springer-Verlag.

[124] William Noah Schilit. *A System Architecture for Context-aware Mobile Computing.* PhD thesis, Columbia University, New York, 1995.

[125] Mark Weiser. The computer for the 21 st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11, jul 1999.

[126] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a Better Understanding of Context and Context-Awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer-Verlag, sep 1999.

[127] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The context toolkit: Aiding the Development of Context-enabled Applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*, pages 434–441, New York, New York, USA, may 1999. ACM Press.

[128] Dana Pavel and Dirk Trossen. Context Provisioning for Future Service Environments. In *2006 International Multi-Conference on Computing in the Global Information Technology - (ICCGI'06)*, pages 45–45. IEEE, aug 2006.

[129] A. Corradi, R. Montanari, D. Tibaldi, and A. Toninelli. A Context-centric Security Middleware for Service Provisioning in Pervasive Computing. In *The 2005 Symposium on Applications and the Internet*, pages 421–429. IEEE, 2005.

[130] Su Sheng, W. L. Chan, K. K. Li, Duan Xianzhong, and Zeng Xiangjun. Context Information-Based Cyber Security Defense of Protection System. *IEEE Transactions on Power Delivery*, 22(3):1477–1481, jul 2007.

[131] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, pages 1–6. USENIX Association, oct 2010.

[132] Earl Oliver. The challenges in large-scale smartphone user studies. In *Proceedings of the 2nd ACM International Workshop on Hot Topics in Planet-scale Measurement - HotPlanet '10*, pages 1–5, New York, New York, USA, jun 2010. ACM Press.

[133] Antti Oulasvirta, Tye Rattenbury, Lingyi Ma, and Eeva Raita. Habits make smartphone use more pervasive. *Personal and Ubiquitous Computing*, 16(1):105–114, jun 2011.

[134] Hannu Verkasalo. Contextual patterns in mobile service usage. *Personal and Ubiquitous Computing*, 13(5):331–342, mar 2009.

[135] Tapio Soikkeli, Juuso Karikoski, and Heikki Hammainen. Diversity and End User Context in Smartphone Usage Sessions. In *2011 Fifth International Conference on Next Generation Mobile Applications, Services and Technologies*, pages 7–12. IEEE, sep 2011.

[136] Michael Armbrust, Ion Stoica, Matei Zaharia, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, and Ariel Rabkin. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, apr 2010.

[137] Google App Engine: Platform as a Service, 2015. `https://cloud.google.com/appengine/docs`.

[138] Lifei Wei, Haojin Zhu, Zhenfu Cao, Xiaolei Dong, Weiwei Jia, Yunlu Chen, and Athanasios V. Vasilakos. Security and privacy for storage and computation in cloud computing. *Information Sciences*, 258:371 – 386, feb 2014.

[139] R.G. Lennon. Changing user attitudes to security in bring your own device (BYOD) & the cloud. In *Tier 2 Federation Grid, Cloud High Performance Computing Science (RO-LCG)*, pages 49–52, 2012.

[140] Sanjay P. Ahuja and Deepa Komathukattil. A Survey of the State of Cloud Security. *Network and Communication Technologies*, 1(2):66–75, nov 2012.

[141] Zhifeng Xiao and Yang Xiao. Security and Privacy in Cloud Computing. *IEEE Communications Surveys & Tutorials*, 15(2):843–859, jan 2013.

[142] Everaldo Aguiar, Yihua Zhang, and Marina Blanton. An Overview of Issues and Recent Developments in Cloud Computing and Storage Security. In Keesook J. Han, Baek-Young Choi, and Sejun Song, editors, *High Performance Cloud Auditing and Applications*, pages 3–33. Springer New York, New York, NY, 2014.

[143] Siani Pearson. Privacy and Security for Cloud Computing. In Siani Pearson and George Yee, editors, *Privacy, Security and Trust in Cloud Computing*, Computer Communications and Networks, pages 3–42. Springer London, London, 2013.

[144] Minqi Zhou, Rong Zhang, Wei Xie, Weining Qian, and Aoying Zhou. Security and Privacy in Cloud Computing: A Survey. In *2010 Sixth International Conference on Semantics, Knowledge and Grids*, pages 105–112. IEEE, nov 2010.

[145] Luis M. Vaquero, Luis Rodero-Merino, and Daniel Morán. Locking the sky: a survey on IaaS cloud security. *Computing*, 91(1):93–118, nov 2011.

[146] Luis Rodero-Merino, Luis M. Vaquero, Eddy Caron, Adrian Muresan, and Frédéric Desprez. Building safe PaaS clouds: A survey on security in multitenant software platforms. *Computers & Security*, 31(1):96–108, feb 2012.

[147] HP 2012 Cyber Risk Report. HP 2012 Cyber Risk Report. Technical report, HP, 2012.

[148] George Grispos, William Bradley Glisson, and Tim Storer. Using Smartphones as a Proxy for Forensic Evidence Contained in Cloud Storage Services. In *2013 46th Hawaii International Conference on System Sciences*, pages 4910–4919. IEEE, jan 2013.

[149] Infosecurity. Recycled phones retain their previous owners' data, 2013. `http://www.infosecurity-magazine.com/news/recycled-phones-retain-their-previous-owners-data/`.

[150] Qing Li and Greg Clark. Mobile Security: A Look Ahead. *IEEE Security & Privacy*, 11(1):78–81, jan 2013.

[151] YouGov. YouGov / Blackbelt Survey Results. Technical report, YouGov plc, 2013.

[152] Yi Wei and M. Brian Blake. Service-Oriented Computing and Cloud Computing: Challenges and Opportunities. *IEEE Internet Computing*, 14(6):72–75, nov 2010.

[153] John Domingue and Dieter Fensel. Toward a service web: integrating the Semantic Web and service orientation. *IEEE Intelligent Systems*, 23(1):86–88, nov 2009.

[154] Marco Vieira, Nuno Laranjeiro, and Henrique Madeira. Benchmarking the Robustness of Web Services. In *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, pages 322–329. IEEE, dec 2007.

[155] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-Oriented Computing: State of the Art and Research Challenges. *Computer*, 40(11):38–45, nov 2007.

[156] Qi Yu, Xumin Liu, Athman Bouguettaya, and Brahim Medjahed. Deploying and managing Web services: issues, solutions, and directions. *The VLDB Journal*, 17(3):537–572, may 2008.

[157] Quan Z. Sheng, Zakaria Maamar, Lina Yao, Claudia Szabo, and Scott Bourne. Behavior modeling and automated verification of Web services. *Information Sciences*, 258:416–433, feb 2014.

[158] Fengming Liu, Li Wang, Lei Gao, Haixia Li, Haifeng Zhao, and Sok Khim Men. A Web Service trust evaluation model based on small-world networks. *Knowledge-Based Systems*, 57:161–167, feb 2014.

[159] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[160] Fuguo HUANG. *Web Technologies for the Internet of Things.* PhD thesis, Aalto University, 2013.

[161] John Lowry. Designing for Scale, 2013. `https://cloud.google.com/appengine/articles/scalability`.

[162] Quotas, 2014. `https://cloud.google.com/appengine/docs/quotas`.