TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABREVIATIONS

ACL	Access Control List
ABE	Attribute Based Encryption
AES	Advance Encryption Scheme
API	Application Programming Interface
BLE	Bluetooth Low Energy
CP-ABE	Ciphertext Policy Attribute Based Encryption
DCM	Data Collector Module
DM	Decryption Module
DSA	Digital Signature Algorithm
DTLS	Datagram Transport Layer Security
E2E	End to End
ECC	Elliptic Curve Cryptography
ECDDH	Elliptic Curve Decisional Diffie-Hellman
ECIES	Elliptic Curve Integrated Encryption Scheme
EM	Encryption Module
FISMA	Federal Information Security Management Act
FTC	Federal Trade Commission
GDPR	General Data Protection Regulation
GEM	Gateway Encryption Module

IoT	Internet of Things
KG	Key Generation Module
KP-ABE	Key Policy Attribute Based Encryption
LoWPAN	Low-Power Wireless Personal Area Networks
MCC	Mobile Cloud Computing
MSK	Master Secret Key
MQTT	Message Queuing Telemetry Transport
OSGi	Open Service Gateway initiative
PBC	Pairing Based Cryptography
PEM	Proxy Encryption Module
PII	Personal Identification Information
PK	Public Key
PM	Privacy Module
RND	Random Number Generator
RSA	Rivest-Shamit-Adleman Cryptosystem
SAML	Security Assertion Markup Language
SHA	Secure Hash Algorithm
SK	Secret Key
SOAP	Simple Object Access Protocol
SPI	Sensitive Personal Information

TLS	Transport Layer Security
WSN	Wireless Sensor Network
XACML	eXtensible Access Control Markup Language

LISTE OF SYMBOLS AND UNITS OF MEASUREMENTS

T_{act}	Actual Access tree
T_{dum}	Dummy Access tree
α	Access policy
ω	List of attributes
CT_{Dummy}	Dummy Ciphertext
CT_{Actual}	Actual Ciphertext
sec	Seconds

INTRODUCTION

Security and Privacy of Internet of Things (IoT) has become an important issue. Since IoT is becoming one of the most prominent trend in the current technological advancement, as the world is becoming inter-connected with the internet. IoT has influenced all the technological domains starting from a simple fitness tracker to a fully automated autonomous car manufacturing industry. In this chapter we will show the background of IoT, Security and privacy issues concerning IoT and its interleaved technologies, objective of the dissertation, and a brief discussion on the proposed methodologies.

0.1 Internet of Things

According to (Roberto Minerva & Rotondi (2015)), IoT is a system of collaboration of computational elements controlling physical entities with respect to some events in the real world. In different context IoT can be defined as Machine-to-Machine (M2M) communications, where different devices communicate with each other in order to fulfill a common goal together. With the help of IoT, physical objects can be controlled or sensed remotely from anywhere using the existing network infrastructures. The network infrastructure allows direct integration of physical standalone things into computer based systems, which is improving efficiency, accuracy and also reducing human intervention. IoT can be traced back to simple Wireless Sensor Network (WSN) and have evolved to become a sophisticated and complex mesh network of devices, domains, and industries as shown in Figure 0.1.

Some of the examples of IoT are in health-care, smart-grid, self-driving cars and drones, video surveillance system with tracking using image recognition, contactless and biometric systems for payment, agriculture and mining, production and even in education and training (Ashton (2016)). IoT has influenced in every aspects of personal life, ranging from automated coffee

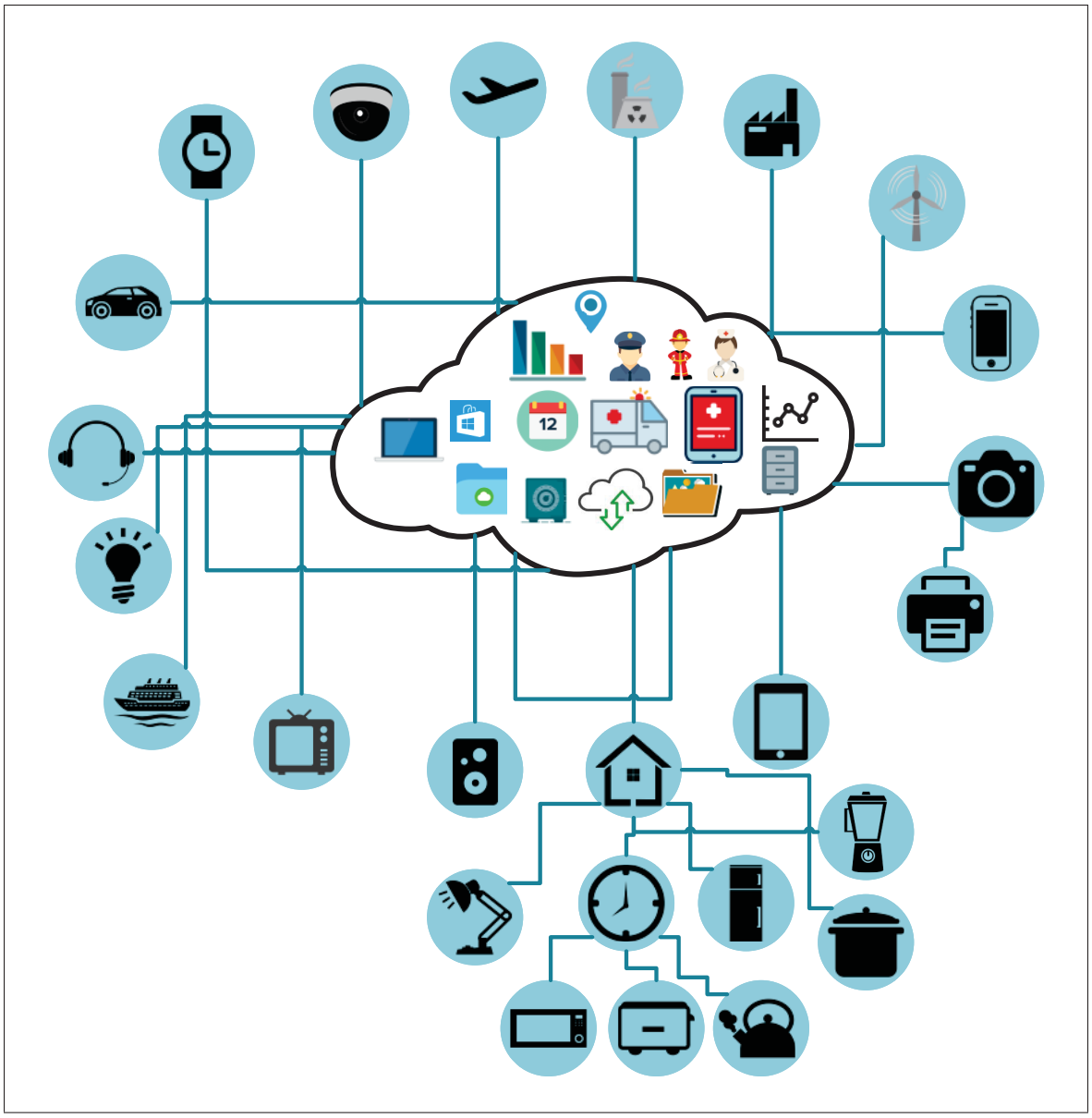


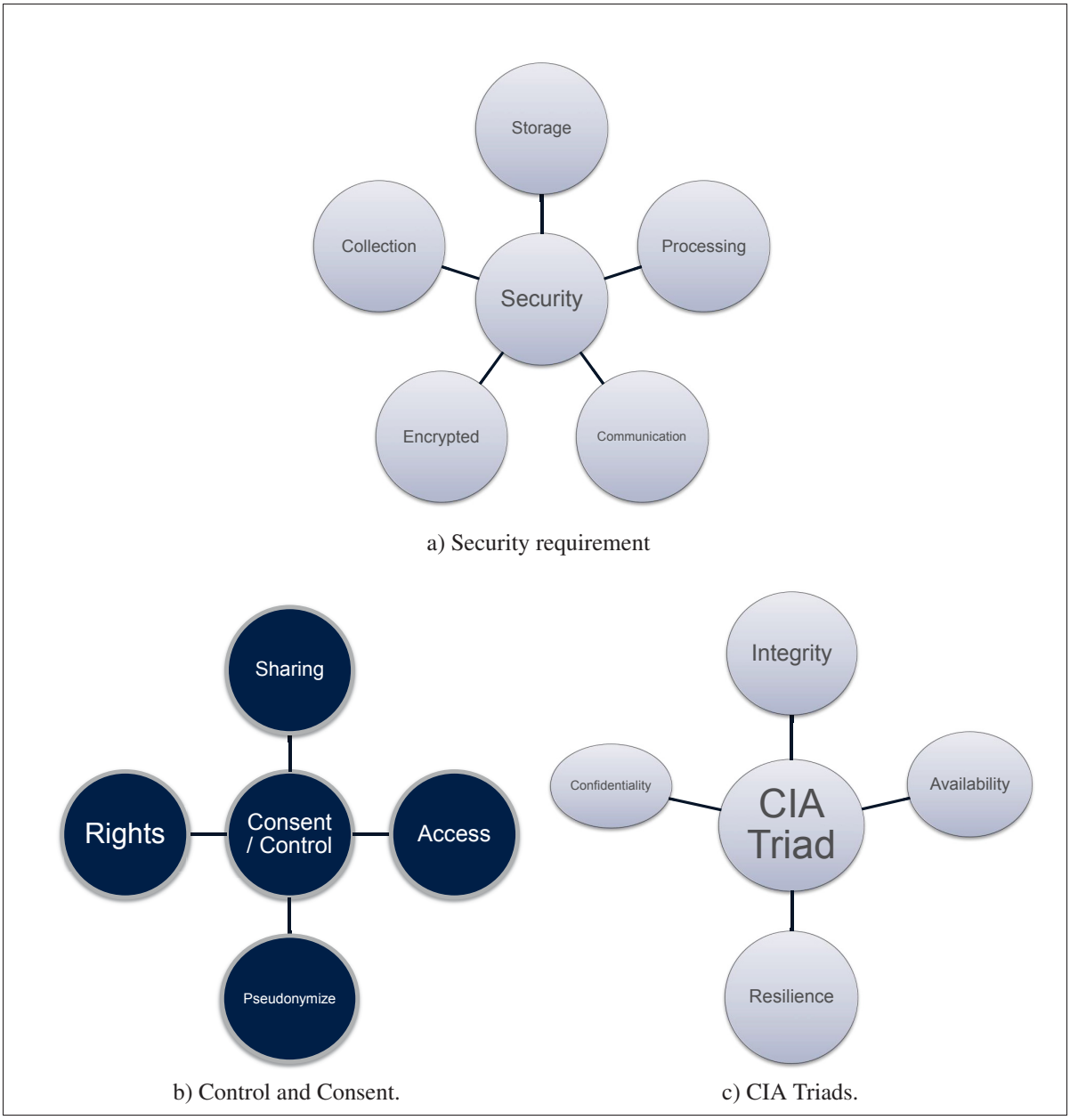
Figure 0.1 IoT Overview.

machine to a fully automated smart environment or cities. The trend IoT is following (Newman (2017)), it will become more fragmented and distributed.

0.2 Security issues regarding IoT

Security and privacy always been a challenging aspect while dealing with technology and personal data. Due to the rapid technological advancement, the need for security and privacy has increased drastically over the few years. According to the Annual Cybercrime Report (Morgan (2018)) from Cybersecurity Ventures predicts that cybercrime will cost $6 trillion annually by 2021. Meanwhile, Ponemon Institute funded by IBM (Institute (2018)), conducted a study on Data Breach that the global average cost of a data breach is $3.62 million. According to Gartner (Gartner (2017b)), worldwide spending on information security products and services at $86.4 billion in 2017, and they predicted that spending will increase to $93 billion in 2018.

Data Confidentiality, Integrity and Availability (CIA) Triad (Figure 0.2c) is the basic model for designing any technological security system. CIA Triad allows the security experts and manufacturers to balance the different aspects aspects for the emerging technologies. As the cyber threat vectors are becoming more complex day by day there is an addition of "resilience" in the CIA triad. Resilience means failure of any discrete component should not cause systemic failure. According to the new laws and regulations enforced by the government like FTC (FTC (2016)), GDPR (GDPR (2016)), FISMA (FISMA (2016)) for personal data protection, as for example, the guideline for protecting PII (Erika McCallister & Scarfone (2010)), the security aspects while dealing with personal data has to changed. Figure 0.2a and Figure 0.2b shows the security and privacy requirements for dealing with the personal data, for example, all data should be kept encrypted, the communication has to be secured as well processing of data, the data owner has the right to decide what part of the data can be accessed by the service provider and what should be shared to the third party, etc. Also the regulations state that the service providers have to protect personal data during their business period and then delete the data after that as well as the service provider can not share those data with any third parties.



a) Security requirement

b) Control and Consent.

c) CIA Triads.

Figure 0.2 Security Aspects.

The popularity of IoT is increasing mainly due to the drastical increase of popularity and crave for smart devices, sensors, cheaper devices and the capabilities of cloud computing. IoT devices fully depend on the computational power of the cloud and existing network infrastrucutures to work together as one to serve a common purpose. The services provided by the Cloud can easily satisfy the massive and changing demand on different resources constrained devices.

It is providing all the necessary services like computation, analytics and storage for the IoT resource constrained devices (Avoyan (2017)). According to (McKendrick (2016)) by the end of 2020, 68% of the cloud workloads will be in public cloud data centers. So data privacy and security in the third party cloud services brings new concerns, since the clouds might be honest but as well as curious (Chai & Gong (2012)) by collecting information without notifing the data owner. There are several security issues (Brodkin (2008)) (Prinzlau (2017)) related to the third party cloud service, for example, there are chances of for the loss of sensitive data and data leakage which raises the risk of data misused by the service provider or attacker, cloud credential and key management of cloud services might lead to potential breaches. Al Morsy *et al*. (Almorsy *et al.* (2016)) showed some of the security issues relevant to virtualization, multi-tenancy, management and hostile networks. Even though cloud services provides basic security features and functionality to ensure the security of the whole database, but most of the time the data itself is not secured, so there is always a single point of failure.

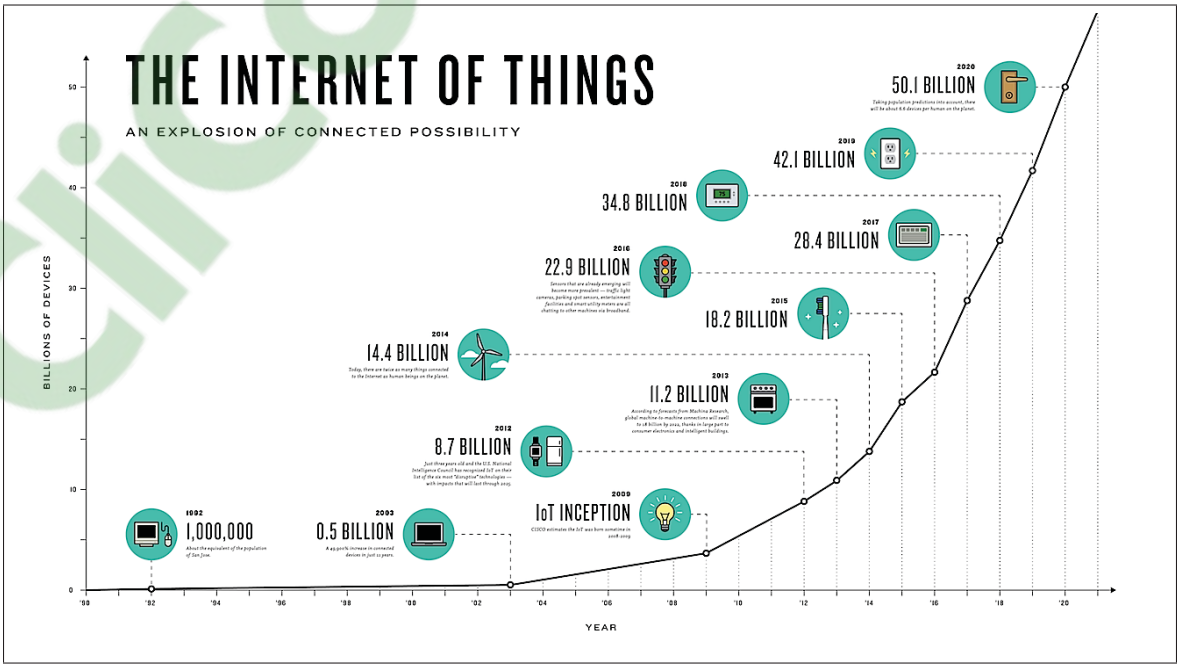


Figure 0.3 IoT prediction. (Ródenas (2015))

According to (Gartner (2017a)), there are 8.4 billion connected things in 2017 and by the end of 2021 there will 50 billion devices (Peter Middleton & Rueb (2017)) as shown in Figure 0.3. Most of the time IoT devices send data to the cloud which are usually small in size but the amount of data being sent are usually periodical, which eventually leads to a huge volume. From a survey conducted by Cisco (McKendrick (2016)) (Cisco (2017)) that IoT will generate 600 Zetabyte of data by 2020 which is an 275% increase from 2016. Data coming from the sensors and devices in domain of smart-home or healthcare and etc. contain Personal Identification Information (PII) or Sensitive Personal Information (SPI). PII (Wikipedia (2018)) is a term used to distinguish or trace an individual's identity and habit, such as name, social identification number, date and place of birth, when the person sleeps, daily routine and etc. If these PII or SPI is either misused by service provider or any third party or even there is a breach in the cloud where the data is being stored there will be serious consequences on the security and privacy of the data owner. Access control of the data is the right of the data owner, the owner should decide which data to be shared, what are not and to whom the data will be shared. There are few methods of data access control like Role-Based Access Control (Ferraiolo *et al.* (1995)) and Attribute-Based Access Control (Yuan & Tong (2005)) but they are not computationally adequate to run on IoT devices and also these techniques will totally block the data from sending to the cloud and sometime it is important to keep the history of the data for the future in case of emergency or for archiving as all data are important.

In the context of security in IoT devices are the most challenging, since regular security mechanisms cannot be incorporated, as IoT devices does not have the adequate resources to perform complicated encryption or to integrate security and privacy mechanisms, so in most of the cases IoT devices are vulnerable for being exploits. Different types of experiments or exploits were performed with the IoT environment using commercial of-the-shelf products. One of this type of experiment (Schurgot *et al.* (2015)) is done using SMART Things which is a popular middleware for smart home. In this experiment, there were mainly privacy attack on system

and are performed using cryptographic technique and information manipulation. According to a survey by eclipse (Skerrett (2017)), security is one of the key concern for IoT which includes communication security, data encryption, PKI and etc. Also a report by AT&T (AT&T (2016)) showed that there is a 458% increase in vulnerabilities in IoT devices in the two years. Some of worst IoT exploits are Mirai Botnet (Antonakakis *et al.* (2017)), hackable cardiac pacemakers (Osborne (2017)), Owlet baby heart monitor vulnerabilities (Stanislav & Beardsley (2015)), TRENDnet webcam hack (Zetter (2012)), smart home devices used as botnet (BBC (2017)) and etc. The vulnerabilities and exploits in IoT are happening are mainly due to the lack of light weight security mechanisms which can be used in these resource constrained devices (Maddox (2016)). In addition, most of the security mechanism deals with E2E secured connection like TLS, DTLS, etc. but there are few security mechanism that deals with the data security, integrity and access control. All the current technologies in the IoT domain provides End-to-End (E2E) security but lacks adequate security mechanisms incorporated with them especially data privacy, data encryption and access control (Barnes (2017)).

0.3 Objective

In this project, we are mainly focusing on one of the security aspect of IoT regarding the data privacy and integrity at the source using Smart Home as our case study. Smart home (Figure 0.4) is one of the IoT applications that is leveraging the Cloud adequately. Since most of the sensors and devices do not have enough computational power and are resource constrained, so outsourcing the data to the cloud is one of the best solution for data storage, analytics and computation. Additionally, connecting the smart home to the cloud allows analytics which tempts many services like healthcare, surveillance, assisted living, smart grid and etc.

Security and privacy has become an important issue in a smart home environment, since sensors and smart devices are pervasively sends data to the cloud which can lead to information that can reveal a person's daily routine (Wilbanks (2007)). As discussed before, outsourcing sensed

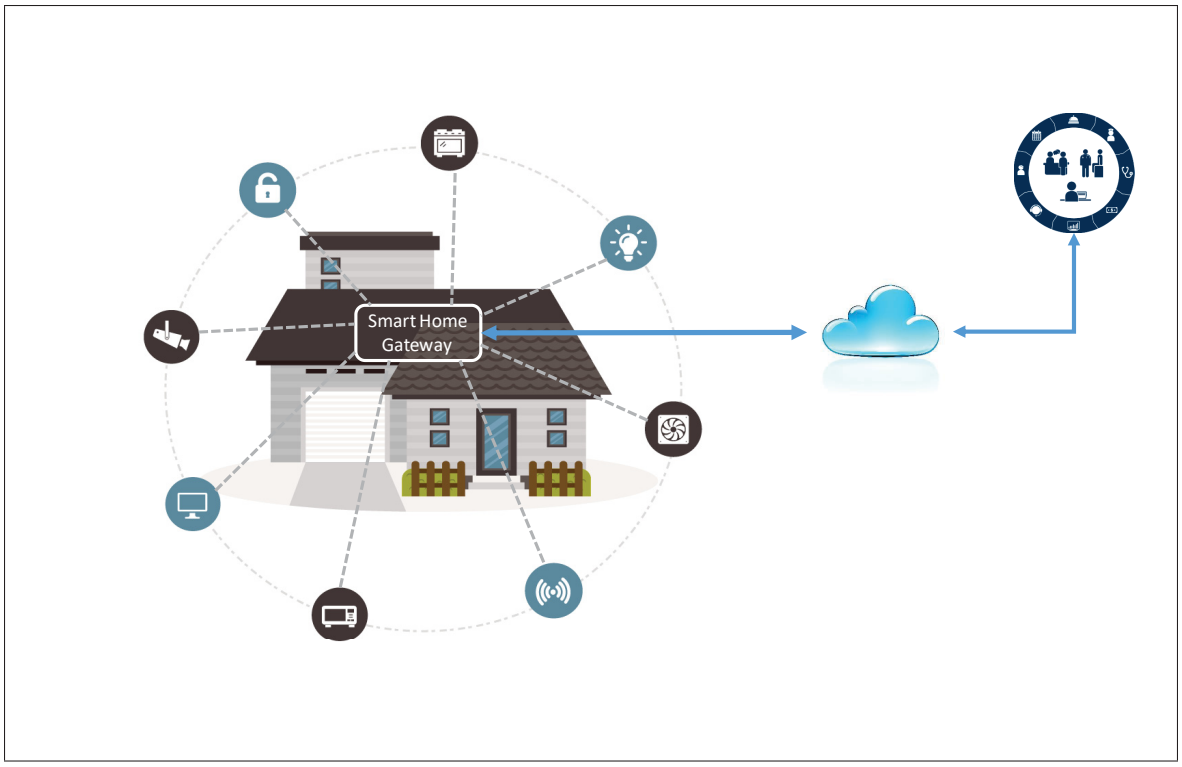


Figure 0.4 Generic Smart Home.

data from smart home to the cloud will cause serious privacy concerns. One privacy preserving approach addressing this specific problem consists to encrypt data, before its sent to the cloud, according to smart home owner preferences. The main objective of this research is to add or incorporate a suitable security integration methods for privacy preserving mechanism in an IoT environment specifically Smart Home.

0.4 Methodology

In order to provide security and privacy, we are proposing a cryptographic encryption approach where users' can access data which is granted by the home owner only. For this purpose, we are proposing two architectures integrating Attribute-Based encryption (ABE) (Goyal *et al.* (2006)) schemes in the Smart Home Middleware. ABE is an asymmetric encryption scheme where data is encrypted and decrypted using some attributes like user id, service provider,

sensor category, etc. The data owner has the overall access control of the data by specifying an access policy over finite number attributes. Only authorized users owning the required attributes are required to satisfy the policy can decrypt the ciphertext. There are mainly two schemes belonging to ABE class are: Key-Policy Attribute-Based Encryption (KP-ABE) (Goyal *et al.* (2006)) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) (Bethencourt *et al.* (2007)). The main advantage of ABE is the possibility of specifying flexible fine-grained access control policies over encrypted data, which is an important requirement for the privacy of the data. In addition, this encryption system does not put any restriction on neither the number of authorized entities nor their identities. This feature enables a reliable anonymous access control. Our proposition is providing architectures and techniques to provide data security, integrity and access control from the source even in the cloud, so if the cloud or data is compromised the attacker will not have access to the plaintext data.

We incorporated ABE mechanism into an well-known IoT middleware 'openHAB' (openHAB (2018)) (Chapter 3) as a plugin for portability. In order to show that the security architecture is feasible, we did simulation of a smart home with multiple sensors, users and service provider. Finally we evaluated different ABE schemes to see the performance of the architecture based on resource consumption, latency, frequency of data and the number of sensors with the middleware.

Then we extended the architecture by offloading the ABE encryption to a proxy server (Chapter 4), to reduce the overhead in the smart home gateway. The main idea of this implementation is to do partial encryption at the smart home gateway and rest of the encryption is passed to a proxy server in order to reduce the computational overheads on the resource constrained device. In this architecture we also designed a data collector which acts as a middleware to reduce extra computational overhead of openHAB. Finally we evaluated different ABE schemes using the

offloading technique to see the performance of the architecture based on resource consumption, latency and interval of data.

0.5 Contributions

The purpose of this research is to provide a mechanism for data privacy in an IoT middleware. The research is mainly divided in two parts, firstly privacy mechanism using an open-source IoT middleware and later an optimization of the ABE scheme using offloading technique. The outline of the contributions of our research are as follows:

- An integration mechanism for privacy preserving technique using ABE with a well known IoT middleware 'openHAB';
- Simulating a smart home to show that the mechanism is feasible for a smart environment;
- Experimentation with various ABE techniques to provide a comparison of different ABE schemes with respect to resource consumptions and latencies.
- Extending the privacy preserving technique by encryption offloading using dummy attribute concept;
- Implementation of a simple data collector which serves as middleware to see the performance of the scheme;
- Experimentation with various ABE techniques using the concept of dummy attribute for offloading encryption on different ABE schemes with respect to resource consumptions and latencies.

0.6 Summary

The rest of the dissertation is organized as follows, Chapter 1 presents the backgrounds of middlewares and cryptographic preliminaries and schemes, Chapter 2 reviews the related works, Chapter 3 and Chapter 4 present the architectures, implementations and experimentations, Chapter 5 presents the future work, and we summarize the dissertation in the Conclusion.

CHAPTER 1

BACKGROUND

The purpose of the background study presented, is to provide information regarding the encryption preliminaries and schemes that is being used in this dissertation as well as the backgrounds of IoT middlewares. The chapter begins with the IoT middlewares, middleware requirements, types of middlewares and some examples of IoT middlewares specifically designed for smart home. Then we showed the comparison of different middleware using some predefined criteria. Then we moved to the basic idea of pairing based cryptography, ABE schemes, types, and algorithms.

1.1 IoT Middleware

Middleware or Smart Gateway plays an vital role in the IoT infrastructure. Middleware is a collaboration of hardware and software which is responsible for receiving or sending the data from and to the sensors on behalf of the users. It serves as an intermediary for the embedded systems and the application to communicate with each other. They also provide a platform for the users or services to communicate with the sensors and actuators. Some of the middleware requirements for IoT are functional, non-functional and architectural (Razzaque *et al.* (2016)) (Stankovic (2014)).

- **Functional requirements**

  - **Resource discovery**: The middleware must have mechanisms to detect the presence of devices or sensors and should be able to dynamically connect to them.
  - **Resource management**: A middleware needs the manage the services provided by the sensors or devices and should be able to monitor them.
  - **Data management**: A middleware needs to have the capabilities to manage data that is sent to it from the sensors or devices for processing, filtering and storage.

- **Event management**: A middleware should be able to handle huge number of events that is being sent from sensors or devices without creating congestion or degrading the performance of the system.
- **Code management**: A middleware must be able to deploy codes that is upgrade version of the application without being interrupted from its regular services.

- **Non-functional requirements**

- **Scalability**: A middleware needs to have be scalable to the growth of the devices or sensors so that it can accommodate the applications and network.
- **Real time**: The middleware should provide services in real-time and on time when an event is occurred.
- **Reliability**: The middleware should be reliable and work smoothly during the lifetime of the system even if there is a failure.
- **Availability**: The middleware has to available or appear to be online especially in mission critical situation like in healthcare. The middleware should be prune to fault tolerance.
- **Security and privacy**: The middleware must have security mechanism so that there no malicious attacker can have access into the middleware is not information leakage.
- **Ease-of-deployment**: The middleware needs to have the portability that is easily deployed with less or no knowledge of the system.
- **Popularity**: The middleware should provide service and support throughout the life time of the system.

- **Architectural requirement**

- **Programming abstraction**: The middleware needs to provide API for developers for faster development of application.
- **Interoperable**: The middleware has to work with heterogeneous devices, technologies or application without additional effort.

- **Service-based**: The middleware should be service-based to offer high flexibility when a function needs to be added or deleted.
- **Adaptive**: The middleware needs to adaptive so that it can evolve to fit itself into changes in its environment.
- **Context-aware**: The middleware should be aware of the users, devices or environments context and use these for effective and essential services offering to users.
- **Autonomous**: The middleware must be able to communicate with each other without human intervention.
- **Distributed**: The middleware has to be sufficient to support many distributed services and application.

1.1.1 Different kinds Middleware

There are different types of middlewares, and they are categorised as follows (Razzaque *et al.* (2016))

- **Event-based middleware**: This type of middleware uses change in state for interaction, that is publish/subscribe between the embedded systems to the cloud.
- **Service-oriented middleware**: This kind of middleware uses service based subscription for transfer of data from the cloud to the embedded system.
- **VM-based middleware**: VM-based middleware uses virtualizations of embedded systems in the cloud. Each embedded system has its own image in the middleware and the users communicate with the image for getting the data from the sensors.
- **Agent-based middleware**: These middleware view the embedded systems as agents.
- **Tuple-space middleware**: This type of middleware each sensors and embedded systems are viewed as tuples of a whole system.

- **Database-oriented middleware**: These views the whole system of embedded systems and sensors as a part of a database, each components acts a record of the database.
- **Application-specific middleware**: This middleware is based on specific applications of a domain.

1.1.2 Examples of Middlewares

There are lot of middlewares emerging due to necessity and demands of Do-It-Yourself (DIY) projects. Some of the middleware are open-source and some are paid services. Among the open-source middlewares some of the well-known are as follows:

1.1.2.1 openHAB

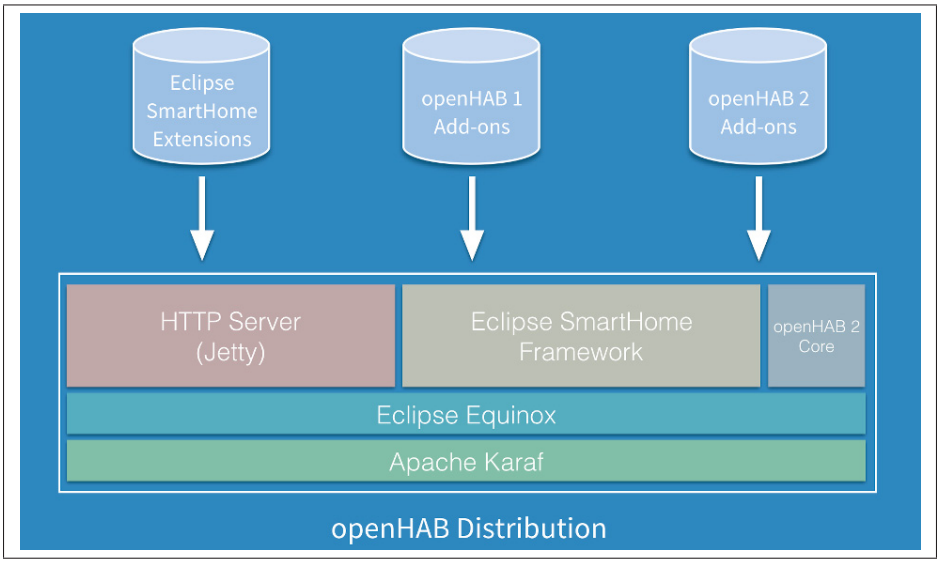


Figure 1.1 Architecture of openHAB) (openHAB (2018))

openHAB (openHAB (2018)) is a software for implementing different home automation systems and technologies into one single platform like SmartThings, Logitech, Harmony Hub, Helios, etc. openHAB is an event-based middleware where the user subscribes to devices notification to get access to the device. openHAB runtime is implemented using Java and is

mainly based on Eclipse SmartHome framework with Apache and Eclipse Equinox for the Open Service Gateway initiative (OSGi) runtime environment. There are two different internal communication one is The Event bus and Item repository. The architecture of the openHAB middleware is shown in Figure 1.1. This middleware doesn't support any security mechanism for secured communication and there is no privacy policies to govern the access to the devices in the smart home.

1.1.2.2 Mbed

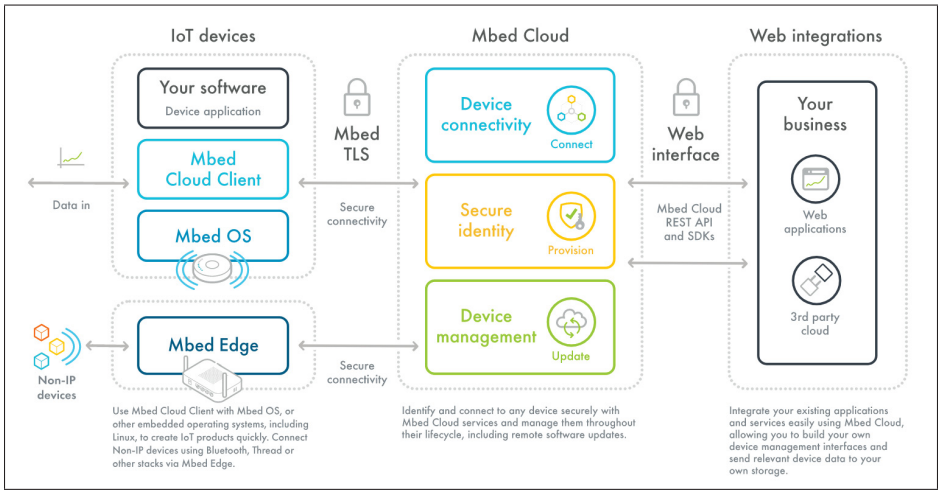


Figure 1.2 Architecture of Mbed (Mbed (2018))

Mbed (Mbed (2018)) is one of latest IoT architecture platform developed by ARM. This platform is specially designed for ARM based microcontrollers and designed for all open standards for connectivity and device management. As shown in Figure 1.2 the architecture is divided into two parts, Mbed OS and the Mbed Cloud. The platform has the ability for device management, device identity as well as integrated security measurement like TLS for secured connection.

1.1.2.3 HomeGenie

HomeGenie (HomeGenie (2018)) (Figure 1.3) is an open source event-based middleware designed on a multi-standard basis. HomeGenie can be interfaced with various devices running on protocols such as Z-Wave, Philips Hue, UPnP / DLNA etc. to communicate with external web services and integrate all of this into a common automation environment.

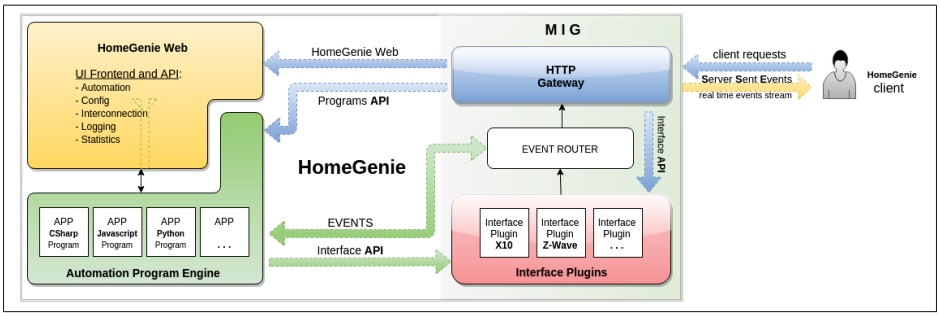


Figure 1.3 Architecture of HomeGenie (HomeGenie (2018))

1.1.2.4 Home Assistant

Home Assistant (Assistant (2018)) (1.4) is open source middleware for home automation. Home Assistant is a message-oriented agent-based middleware were each device act as a component which are added easily and the middleware listen for different type of notifications which is either a trigger or a stream of message. Home Assistant can be extended by components. Each component is responsible for a specific domain within Home Assistant. Component listens for or triggers a specific events, offer services and maintain states. Home Assistant also allows automation of devices.

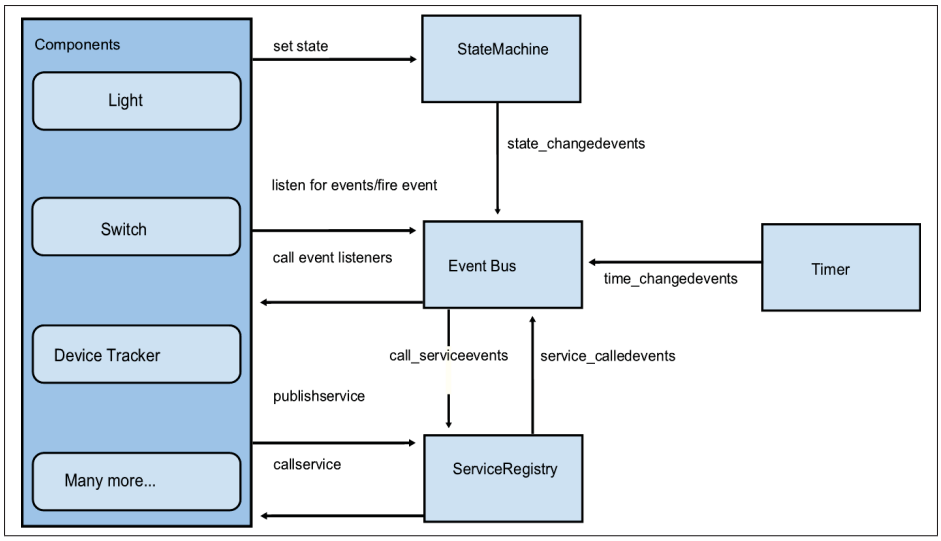


Figure 1.4 Architecture of Home Assistant (Assistant (2018))

1.1.2.5 openRemote

OpenRemote (OpenRemote (2018)) is an open source middleware for home and commercial building automation. Its architecture allows autonomous and user-independent intelligent buildings, some of the products uses openRemote are Philips, ooma, neeo and etc.

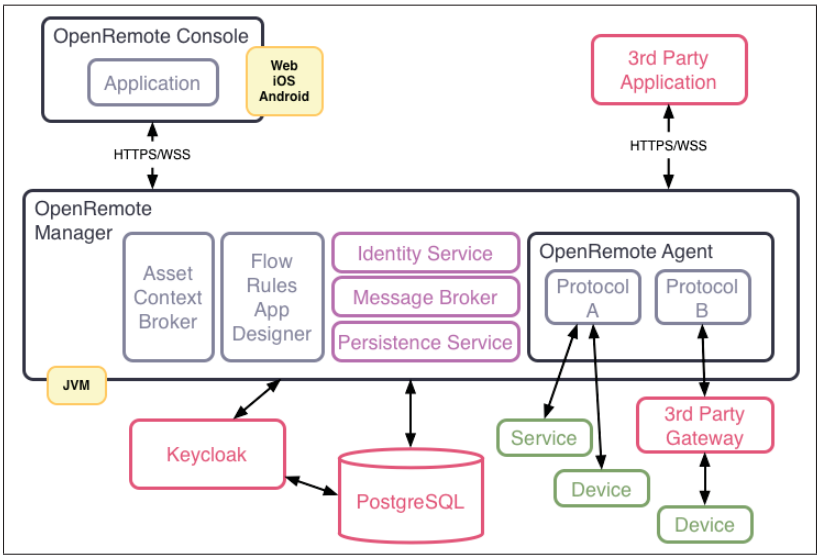


Figure 1.5 Architecture of openRemote (OpenRemote (2018))

1.1.3 Middleware comparison

The comparison of off-the-shelf middlewares are summarized in Table 1.1 based on some specific criteria like supported OS, hardware and security mechanisms available in them. As shown in the table, all the middlewares support most of the communication protocols like MQTT, BLE, Zigbee and etc. The middlewares under survey supports all the Operating systems available except for the Mbed which has its own architecture and OS requirements. From the table we can see that most of the middleware does support E2E secured connections like SSL, TLS and DTLS but none of them have built-in mechanism for data encryption, data integrity and access control for the privacy of the data. Mbed supports security but the security implementation are limited to specific cryptographic algorithms and certificates.

Table 1.1 Comparison of middlewares.

Middleware	**Type**	**Supported OS**	**Embedded hardware**	**Supported Protocol**	**Security and Privacy**
openHAB	Open Source	Windows, Linux, MacOS, Raspbian	Raspberry Pi, Beagle Bone etc	KNX, Z-Wave, ZigBee,MQTT, Bluetooth, UPnP	SSL No privacy module Basic authentication
HomeGenie	Open Source	Windows, Linux, MacOS,	Not Applicable	X10, Z-Wave, ZigBee UPnP	SSL No privacy module Basic authentication
Home Assistant	Open Source	Windows, Linux, MacOS, Raspbian	Raspberry Pi	Z-Wave,IFTTT ZigBee,MQTT, Bluetooth, etc	SSL No privacy module Basic authentication
OpenRemote	Open Source & Paid Service	Windows, Linux, MacOS,	Not Applicable	KNX, Z-Wave, ZigBee,MQTT, etc	SSL No privacy module Basic authentication
The Thing System	Open Source	Raspbian, Linux, MacOS,	Raspberry Pi, Beagle Bone etc.	RFID, Z-Wave, ZigBee,MQTT, UPnP, etc	TSL No privacy module Basic authentication
Mbed	Open Source	mbed os	ARM cortex, Mbed Boards etc	BLE, Z-Wave, ZigBee,MQTT, Bluetooth, LoWPAN, etc	SSL/TLS Crypto libraries X.509 certificates

1.2 Cryptographic premitives

1.2.1 Pairing-based Cryptography

Pairing-based Cryptography (PBC) is a form of cryptography where a pairing of vectors is used to generate the cryptographic parameters. In order to satisfy the PBC requirements, the pairing group has to satisfy, bilinearity, non-degeneracy and computability. A pairing is a bilinear map function where elements of two vector spaces are combined to form a third element, e.g. G_1 x $G_2 \rightarrow G_T$.

1.2.2 Elliptical Curve Cryptography

Elliptical Curve Cryptography (ECC) is an asymmetric key cryptography based on the elliptic curves structure over a finite field. Elliptical curve is a curve defined as $y^2 = x^3 + ax + b$ (WolframMathWorld (2018)). ECC generates keys based on the elliptical curve equations instead of using large prime numbers which is the traditional method of generating the keys for the cryptographic operations. ECC key generation is faster, smaller in size and are more efficient than PKI cryptographic keys. ECC have the advantage of using smaller key to provide the same level of security compared with non-ECC cryptography.

1.2.3 Access tree

Access tree is the representation of the access policy used in ABE into a tree form structure where attributes of the policy is presented as the leave nodes and the operators are assigned as the non-leaf nodes. For example, if we have a policy like "Hospital A AND Doctor = Bruce OR Hospital B OR Type = cardiac AND Patient = Natasha" as an access control parameter for a patient's report then the Access tree structure is similar to the structure shown in Figure 1.6.

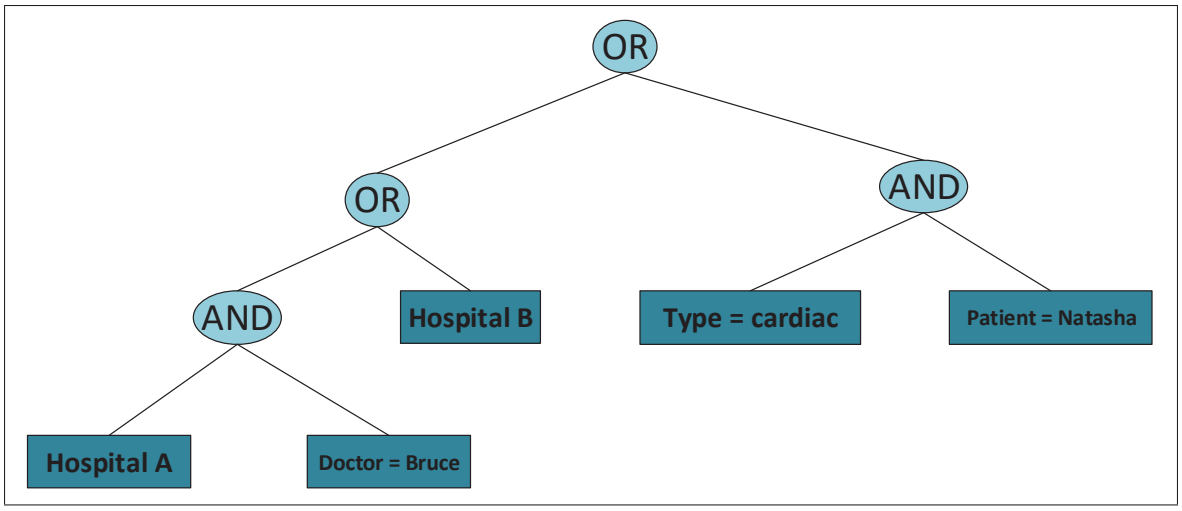


Figure 1.6 Access Tree

1.2.4 Security level

Security level is the measurement for cryptographic algorithms which is related to the key size or key length expressed in bits. Table 1.2 shows the comparison of security level for RSA and ECC (Maletsky (2015)). From the Table 1.2 it is clear that ECC is much stronger than RSA. For example, if we use 1024 bit security of RSA for the key generation, we can achieve the same level of security using only 160 bit of ECC keys, which allows ECC to have same level of security with lower key size compared with RSA.

Table 1.2 Comparison of Key size.

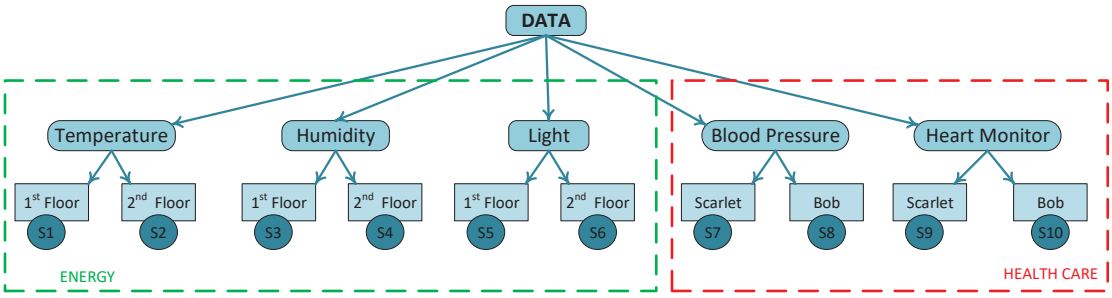
Symmetric Key Size (bits)	**RSA and Diffie-Hellman Key Size** (bits)	**Elliptic Curve Key Size** (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

1.2.5 Attribute Based Encryption

ABE is an asymmetric encryption in which the Secret Key (SK) of the user and Ciphertext (CT) are dependent on a range of attributes (e.g. department, postal code, designation and etc.). ABE was first proposed as an application of fuzzy identity-based encryption where data is encrypted using individual identity defined by a set of attribute (Sahai & Waters (2005)). Later (Goyal *et al.* (2006)) explained the application in details using the term Attribute Based Encryption where data is encrypted using some logical expression of attributes, known as access policy such that encrypted data can be decrypted if that policy is satisfied also the scheme is collusion-resistance.

ABE is one of the encryption schemes that allows fine grained access control, which other symmetric and asymmetric encryption schemes do not offer. For example, as shown in Figure 1.7a, if we consider a scenario, where Scarlet and Bob shares a smart house which has multiple sensors for energy and health care. Scarlet lives in first floor of the house and Bob in second floor and they have given access of the sensors to different Service Providers (SP) as shown in Table 1.7b. For example, SP1 (Service Provider 1) requires all data of Scarlet, in traditional symmetric encryption the data has to be encrypted with different keys for each sensors (s1, s3, s5) where as in ABE scheme the encryption will be done using a policy (s1 OR s3 OR s5) and only one key is required. SP2 requires all energy data of the house, so the encryption will again require multiple keys for encryption where as ABE will require only one key. When we have a complex case like in SP3 and SP4, Scarlet and Bob has to send specific keys which satisfies the sharing of the data, meanwhile in ABE all they have to do is generate a key using a simple policy that satisfies the access. So using ABE in a scenario where access control of the personal data is the most suitable solution.

According to (Goyal *et al.* (2006)), the authors discussed that are mainly two forms of ABE: Ciphertext-Policy ABE (CP-ABE) and Key-Policy ABE (KP-ABE).



a) Simple scenario

Data required	Symmetric	Attribute Based
SP1: 1st floor activity	s1,s3,s5	s1 OR s3 OR s5
SP2: Energy activity	s1,s2,s3,s4,s45,s6	s1 OR s2 OR s3 OR s4 OR s5 OR s6
SP3: 2nd floor humidity from 2014	s4 (all keys from 2014)	s4 AND time= >2014
SP4: Scarlet Morning Healthcare activity	s7,s9 (specific keys)	time= 6am<12pm AND s7 OR s9

b) Comparison of encryption technique

Figure 1.7 ABE representation

1.2.5.1 CP-ABE

CP-ABE is a form of ABE; where the data owner encrypts the data with a access policy, whereas the secret key contains the attributes. The client who intends to decrypt this data must have a secret key that satisfy the policy in CT. Figure 1.8 show visual representation of CP-ABE. CP-ABE has four steps to perform encryption and decryption are as the follows and the details of the algorithms are available in Appendix I.

Setup $\rightarrow$ (PK, MSK): Setup algorithm takes security parameters to generate Public Key (PK) and Master Secret Key (MSK). PK available for any user and used as input for encryption algorithm. MSK used to generate Secret Key (SK) in key Generation algorithm.

KeyGeneration (PK, MSK, ω)$\rightarrow$ SK: KeyGeneration take the PK, MSK, and ω as input. ω is list of attribute of the user. The output of this algorithm is the secret key SK.

Encryption (PK, M, α)$\rightarrow$ CT: This algorithm the user encrypts his/her data with α where α is the access policy. The output of this algorithm is the CT.

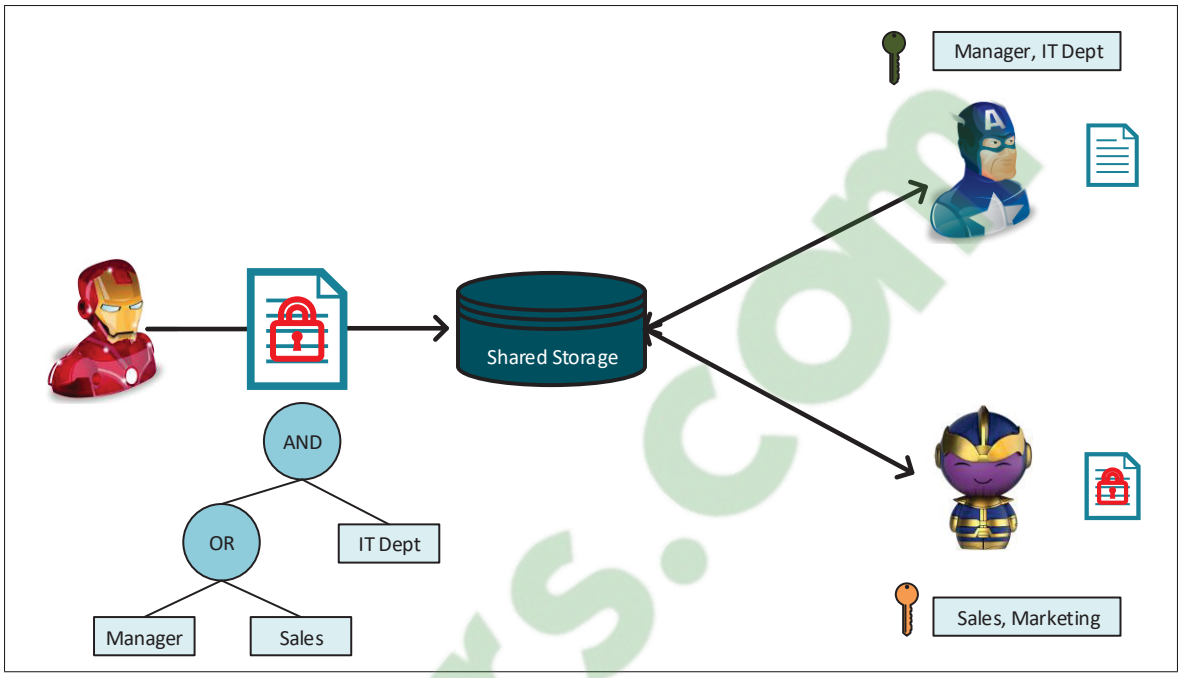


Figure 1.8 CP-ABE

Decryption (CT, SK)→ M: In this algorithm the client uses her secret key to recover the message.

1.2.5.2 KP-ABE

KP-ABE is the second form of ABE; the user encrypts the data with a list of attributes and the secret key incorporates the access policy of the data. The secret key associated with the access policy thus the trust authority who generate the SK will decide who encrypt the data. Figure 1.9 show visual representation of KP-ABE. The following steps explain the main four algorithms of KP-ABE and the details of the algorithms are available in Appendix I.

Setup → (PK, MSK): Setup algorithm used security parameters to generate PK and SK.

KeyGeneration (PK, MSK, α)→ SK: KeyGeneration algorithm used to generate SK. The input of this algorithm PK, MSK, and the α. The algorithm generate SK.

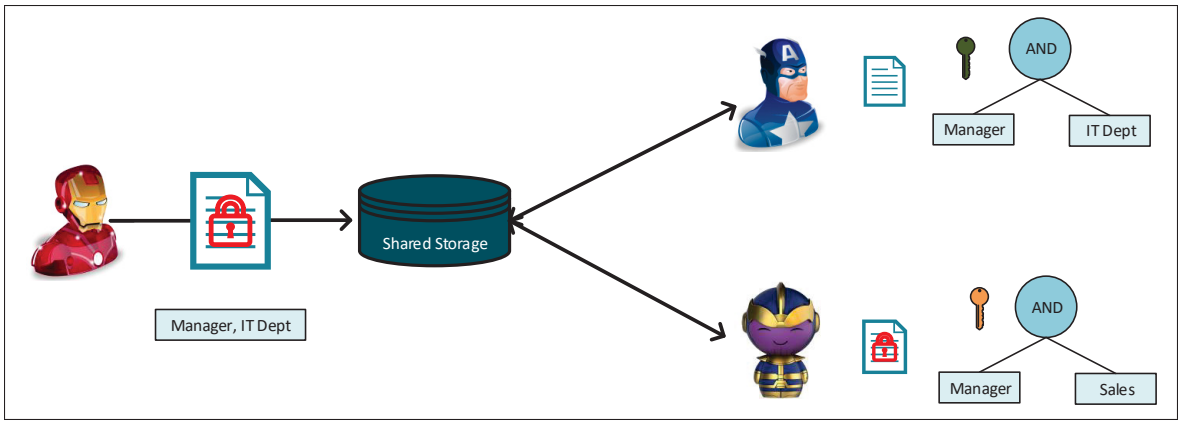


Figure 1.9 KP-ABE

Encryption (PK, M, ω)$\rightarrow$ CT: With this algorithm the user encrypts the data using ω to generate CT.

Decryption (CT, SK)$\rightarrow$ M: This algorithm allows the client to use his/her SK to decrypt CT. If the attributes that the CT associated with satisfy the policy that the SK associated with then the client can decrypt CT and recover the message, otherwise to client will not be able to recover the message.

1.3 Summary

In this chapter, we have shown the basic cryptographic prerequisites for ABE, types of ABE, different algorithms for the ABE schemes. Then we have provided in detailed study of different IoT middlewares, different requirements and presented some examples of open-source middlewares. At the end of the chapter we have shown a comparative study of middlewares based on specific criteria.

CHAPTER 2

LITERATURE REVIEW

There are lot of researches done in the field of IoT and smart home security and privacy. In this chapter we will mainly focus on the security and privacy aspects of IoT and as well as the recent advancement of ABE techniques. We conclude the chapter with the summary of ABE techniques available.

2.1 IoT Security and Privacy

(Singh *et al.* (2016)) analysed twenty security considerations for IoT like secured communications (data leakage and integrity), access control for the IoT cloud (authentication and authorization), identifying sensitive data (PII), encryption at the source and etc. from the perspective of cloud, end-users and the cloud providers. The authors classified the issues as data in transport, identity management, scale of IoT, rise of malicious things, trust, compliance with the regulations and decentralization of IoT and cloud.

(Razzaque *et al.* (2016)) outlined different types of requirements and performed a intensive review of existing middlewares. The authors explained the characteristics of the IoT infrastructure and IoT applications as well as the IoT middlewares requirements like functional, non-functional, architectural and etc. using sixty one well known IoT middlewares. In this article the authors also explained challenges related to the requirements like resource discovery, reliability, security and privacy.

(Henze *et al.* (2014)) designed a trust point-based security architecture for the sensors' data in the cloud which ties the data to the data owners using trust point, which servers as a trusted node between the cloud and the sensors, the trust point has the features to preprocess the data for the cloud and then forwards the data to the cloud for distribution. Also the trust point is responsible for sharing the data key to the data owner. The architecture provides secured E2E communication channel as well as it allows fine grained access control along with key

management. Later they extended the trust point concept using a user-driven enforcement of fine grained policy for the cloud based IoT services (Henze *et al.* (2016)).

(Malina *et al.* (2016)) presented a detailed experimental assessment for the performance of the popular cryptographic algorithms like AES, RND, SHA and RSA on different resource constrained IoT devices like MSP430F149, MSP430F6638, NXP JC3A09002, Nexus 5 and etc. Their experimentation is based on the execution time of the cryptographic algorithm using a constant size message. They have also analysed privacy preserving techniques like k-anonymity, homomorphic encryption, group signatures and etc.

(Punia *et al.* (2017)) provided a summary of the security techniques available in IoT. The article illustrated that a number of researches that is suitable for the IoT case study related to confidentiality, access control, privacy, security protocols and secure routings techniques. The authors also illustrated some of the open issues regarding the physical security of the IoT devices, big data security, application security and nature of IoT heterogeneous networks.

(Fernandes *et al.* (2017)) performed security analysis on one of the most popular IoT framework, SmartThings which is mainly used in smart home environment. They analysed SmartThings products which includes SmartThings app (SmartApp), SmartThings HUB and different smart home devices. They have discovered that 55% of the SmartApp does not provide adequate access control, it has limited security mechanisms and also most of the application does not use all the rights assigned to the devices' operations. From these findings the authors exploited the vulnerabilities and they were able to steal pin codes of the smart lock, access data from the devices, enabling fake fire alarms and etc.

(Jung *et al.* (2011)) implemented a privacy based access control using extensible access control markup language (XACML) and security assertion markup language (SAML). They have used privacy preserving API authorization mechanism, access is given on token based system which is build using SAML and XACML. The architecture has Access token provider which offers a web based user interface to specify and access policy for an application, Access token is provided as SAML assertion containing attributes to the SAML profile of XACML identifying

the enabled soap operation, Access token injector adds the access token to each service request and Access control component acts as a SOAP intermediary and process all SOAP requests to the gateway. This architecture ensures trust using third party policies and data integrity is also ensured. The main disadvantage of this architecture is the packet size of each message is increased from 750 Byte to 14 Kilobyte which increases extra processing for the middleware.

(Marin *et al.* (2007)) proposed a middleware architecture for the smart home using authentication and access control by using credential manager, authentication and trust manager. This middleware is a service-oriented middleware, which provides services and secure access to the devices to the user and applications' data. The middleware uses ACL (Access Control List). This middleware also manages private and context information for flexible device access and control. In this middleware the user request the devices for specific services, then the "Service" contact the "Trust Manager" to check for authorization. Trust Manager guarantees security though the system and gives access right to the users' desired service and also distinguish between users. Trust manager, credential manager and authentication manager uses DSA signature scheme to prevent forgery or tampering. Credential Manager retrieves credentials in form of access control list from the database and gives authorization. Whereas, (Moncrieff *et al.* (2007)) proposed a dynamic adjustable privacy policies framework for smart home based on spatial context, social context, hazard context and activity context. The system uses rule based model and data filtering to ensure the privacy of the user.

2.2 Attribute-Based Encryption

Attribute-Based Encryption is one the best way to achieve security and privacy as well access control using one key for encryption as well as decryption for multiple data based on the access policy of incorporated in the cipher text or the secret key itself. Most of the ABE schemes are usually infeasible for the resource constrained IoT devices mainly due to the execution time and as well as resource consumption for encryption and most of them are not designed for these ubiquitous resource constrained devices. There are different approaches to make ABE schemes lightweight computation for IoT resource constrained devices by performing pre-computation,

removing bilinear pairing which reduces computation but the security level decreases, and etc. In the dissertation, we classify the researches mainly into three categories, firstly feasibility of ABE in resource constrained devices, secondly light-weight ABE encryption for IoT devices and in the last category offloading the encryption to a resourceful devices which performs partial encryption or decryption at the devices and offload the rest of the operations to the cloud.

2.2.1 Attribute-Based Encryption on resource constrained devices

(Ambrosin *et al.* (2016)) experimented on the feasibility of ABE in IoT devices like Intel Galileo, Intel Edison and Raspberry Pi. The experimentation includes execution time, memory utilization and power consumption of CP-ABE scheme for encryption and decryption in different hardwares varying the number of attributes and security level. In the experimentation they have used maximum of 30 attributes and the message size of 3 bytes. Later they used a health-care use case to show the latency of the system for the ABE. (Wang *et al.* (2014)) performed intensive experimentation using ABE schemes on Android smart phone and PC. They evaluated the performance of KP-ABE and CP-ABE schemes using 30 attributes at maximum to find the execution time for the encryption, decryption and key generation based on the security level of ECC curves like 80 bit,112 bit and 128 bit. They also showed the results based on the ECC security level for the ABE and RSA security level.

(Borgh *et al.* (2017)) showed two ways to use ABE in Information-Centric Network (ICN) for the IoT resource constrained sensors. In their first approach, the sensors encrypt the data with symmetric encryption and the keys are shared with the trust authority and then the trust authority encrypts the sensor keys with CP-ABE. The main issue with this approach is that the communication channel needs to be trusted and the management of multiple keys but the advantage of this approach is this requires less overheads on the sensors in terms of computation and resources. On the other approach the sensors encrypts the data with CP-ABE using the PK, which is generated by the trust authority. The main disadvantage of the approach is that it requires more computation as well as execution time and the advantage is that the com-

munication channel does not required to be trusted and also it requires less number of keys for decryption multiple sensor data. They have also performed experimentation based on the number of attributes with respect to the execution time and RAM utilization of the sensors. In their discuss they mentioned that the frequency of the data generated will have an affect the computation.

2.2.2 Light-weight Attribute-Based Encryption for resource constrained devices

(Yao *et al.* (2015)) proposed a lightweight KP-ABE scheme for resource constrained IoT devices by using non-pairing ECC. Their security implementation is based on ECDDH (Elliptic Curve Decisional Diffie-Hellman) for complexity assumption and ECIES (Elliptic Curve Integrated Encryption Scheme) for encryption instead of bilinear Diffie-Hellman based assumption. In their scheme, they did computation during the "setup" algorithm to calculate the parameters with the attributes which saves the time during the encryption process. The main drawbacks of this scheme are poor flexibility regarding revoking or adding attributes, lower scalability and the scheme can not be generalized for any scenario. Whereas, (Oualha & Nguyen (2016)) proposed a ABE scheme by using pre-computation for the CP-ABE, during the pre-computation the scheme stores the expensive ECC settings and the values of the pairs are stored. This technique allows the encryption algorithm to compute the shares faster and eventually the execution of encryption process is reduced. The drawback of this scheme similar to the (Yao *et al.* (2015)).

2.2.3 Outsourcing Attribute-Based Encryption

(Green *et al.* (2011)) are the first researchers to propose an outsourcing technique for ABE. Their scheme provides an efficient and securely decryption of ABE ciphertext. The main change in their scheme is for outsourcing the decryption, which is done using a enhanced version of the "KeyGeneration" algorithm of ABE. There are two keys in this scheme, the first one is a small unique key which is kept by the user (SK), and the second one is a transformation key (TK) which is shared with the cloud as a PK. The TK partially decrypts the CT in

the cloud into a short CT and the user's SK can only decrypt the CT fully. Still, the scheme will cause computational overheads on the encryptor's side and also the decryption will cause overheads as well when there are lot of users. (Qin *et al.* (2015)) proposed an ABE scheme which extends the (Green *et al.* (2011)) implementation by using verifiable outsourced decryption. Their implementation does not increase the computational cost on the users' or clouds sides. Their approach uses a hash function to reduce the size of the message and then there are two symmetric encryptions which provides confidentiality as well as fine grained access control over the data. Also (Lai *et al.* (2013)) extended the implementation of (Green *et al.* (2011)) to provide a solution which allows the users to know if the transformation of the key is done correctly. Whereas, (Balamurugan *et al.* (2013)) focused on the access key structure to improve the security and performance by applying access rights for the authorized users.

(Touati *et al.* (2014)) proposed a C-CP-ABE (Cooperative ciphertext policy attribute-based) where they focused mainly on the encryption algorithm and showed a technique for delegating computational offloading to reduce overhead on resource constrained devices. The main idea of their approach is to delegate the computation of the encryption to the neighbouring unconstrained devices as well as the remote servers. (Ishiguro *et al.* (2013)) proposed key-revocable ABE scheme for the mobile cloud computing. Their research was to reduce the computational overhead cost on the smart devices as well as user revocation and attribute hiding from the cloud server. In order to reduce the computational overhead at the smart devices, they are performing some encryption at the device level and the rest on the server side, so they ended up with nine algorithm (Setup, KeyGeneration, Encypt$_{Usr}$, Encrypt$_{Srv}$, GetCoupon, GetToken, GetMaskKey, Decrypt_Srv, Decrypt$_{Usr}$) for their scheme. The advantage of this scheme that it provides attribute hiding, user can not transfer or copy keys but the main disadvantage of this scheme is the latency, computational cost for decryption and also the feasibility for implementation.

(Zhou & Huang (2012)) proposed a PP-CP-ABE (Privacy Preserving Constant CP-ABE) where the encryption and decryption operations are outsourced to the cloud. According to their finding, the computation of ABE is dependent on the access tree structure and most of the com-

putation for processing the left sub-tree is always higher than that of the right sub-tree. Their implementation uses a specific structure for the access tree so that the left sub-tree has more attributes than the right sub-tree. So they processed the right sub-tree in the resource constrained device and the left sub-tree on a cloud platform. This allows reduced computational processing in the resource constrained devices and more loads to the resourceful devices. Still, the main drawback of this scheme is it not always feasible to generate the trees according to the specific structure and also this scheme is restricted on the flexibility of designing the tree structure. Later, (Jin *et al.* (2015)) enhanced the idea proposed by (Zhou & Huang (2012)) using a dummy attribute. The scheme automatically adds a dummy attribute to the actual attribute list, so the access tree becomes $T = T_{act} \bigwedge T_{dum}$ where T_{act} is the access tree with actual attributes and T_{dum} is the dummy attribute. This technique allows the access tree to have more attributes on the left sub-tree than the right sub-tree. Therefore, during encryption the device just only computes the data using the T_{dum} and the T_{act} is computed in the cloud.

Table 2.1 shows the summary of the ABE schemes based on the constructions, where we showed the different schemes of ABE, type of the scheme, whether the implementation is suitable for resource constrained devices and as well as the scheme is suitable for outsourcing the encryption process.

2.3 Summary

In this chapter we have discussed the different researches doing on in the field of IoT security and privacy. From the related works, it is clear that most of the research are done mainly for the privacy of the IoT domains where they are enforcing different access control mechanism like trust point architecture, XACML or SAML and as well as E2E communication techniques like DTLS. Some of the related works discuss the encryption overheads of resource constrained devices. Whereas, others researched on different ABE schemes and techniques which are applied on resource constrained devices, which can be used for data encryption along with access control. Most of the ABE schemes available shows the feasibility of the schemes with respect to resource constrained devices. But few works and research were implemented with a

Table 2.1 Comparison of different schemes based on the construction.

Scheme	CP-ABE	KP-ABE	ECC	Bilinear Pairing	Constrained Device	Outsource
Bethencourt *et al.* (2007)	✓	✗	✓	✓	✗	✗
Goyal *et al.* (2006)	✗	✓	✓	✓	✗	✗
Yao *et al.* (2015)	✗	✓	✓	✗	✓	✗
Touati *et al.* (2014)	✓	✗	✓	✗	✓	✗
Borgh *et al.* (2017)	✓	✓	✓	✗	✓	✗
Ambrosin *et al.* (2016)	✓	✓	✓	✗	✓	✗
Wang *et al.* (2014)	✓	✓	✓	✓	✓	✗
Ishiguro *et al.* (2013)	✓	✗	✗	✓	✓	✓
Green *et al.* (2011)	✓	✓	✓	✗	✓	✓
Balamurugan *et al.* (2013)	✓	✗	✗	✓	✓	✓
Lai *et al.* (2013)	✓	✗	✓	✗	✓	✓
Oualha & Nguyen (2016)	✓	✗	✓	✓	✓	✓
Qin *et al.* (2015)	✓	✓	✓	✓	✓	✓
Zhou & Huang (2012)	✓	✗	✓	✓	✓	✓
Jin *et al.* (2015)	✓	✗	✓	✓	✓	✓

real IoT environment or middlewares, but these works does not incorporates experimentation with number of devices, the frequency of data generated by the sensors, devices and etc., and also the type of data are not being considered.

CHAPTER 3

ATTRIBUTE-BASED ENCRYPTION FOR SMART HOME

Smart home is one of the IoT applications that is utilizing the full advantage of the Cloud services. Since sensors, actuators and smart devices are resource-constrained and also they are designed for longevity, so outsourcing data to the Cloud for storage, analytics, processing and sharing is the best solution. One of the common privacy issue is being encountered when untrusted third party cloud services and parties are accessing the sensitive data and using these PII beyond their collection purpose. One privacy preserving approach addressing this specific problem is to encrypt data at the source, before the data is sent to the cloud, according to smart home owner preferences. So, incorporating security and privacy in IoT devices is the most challenging due to the nature of these devices. Traditional access control mechanism like XACML and SAML will totally block the data and for and encryption mechanism like symmetric cryptography will generate multiple number of keys for data encryption. So we are proposing to use ABE schemes as our solution model. In this chapter we will be illustrating data security, privacy and access control using ABE cryptography integrated with a popular open-source middleware openHAB.

3.1 Assumptions and Configurations

We are assuming that the data coming from the sensors to the middleware are protected using symmetric cryptography and Transport Layer Security (TCP/IP with TLS and 128 bit AES preshared key). The service providers and the users keys are generated by the owner, which is transferred to the them through secured channel. For the cryptographic parameters we have used supersingular elliptical curve with 512 bit security (SS512) and Miyaji, Nakabayashi and Takano (MNT) curve (Miyaji *et al.* (2000)) with 224 bit security (MNT224).

3.2 Architecture

Figure 3.1 shows an architecture overview of the proposed solution that integrates ABE with openHAB. The system is divided into several modules:

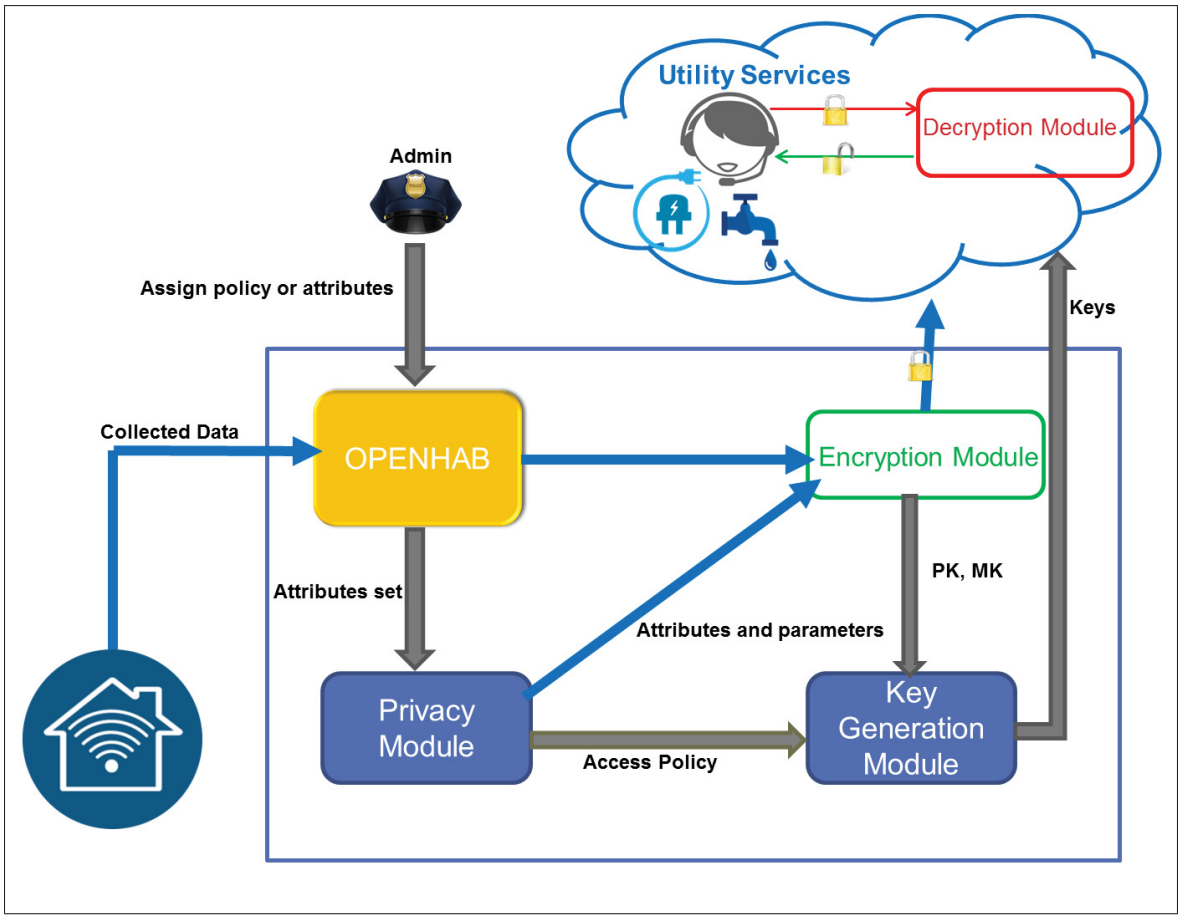


Figure 3.1 Architecture Overview.

OpenHAB is the middleware responsible for data collection from the sensors, providing interface for the home owner to communicate with the PM. It also provides the interface for the decryption module (DM) for the service providers to view the data.

Encryption Module (EM) is responsible for the generation generation of Public Key (PK) and Master Secret Key (MSK) which is required for encryption and KG module. EM is also responsible for the data encryption. The flow chart of the EM is shown in the Figure 3.2.

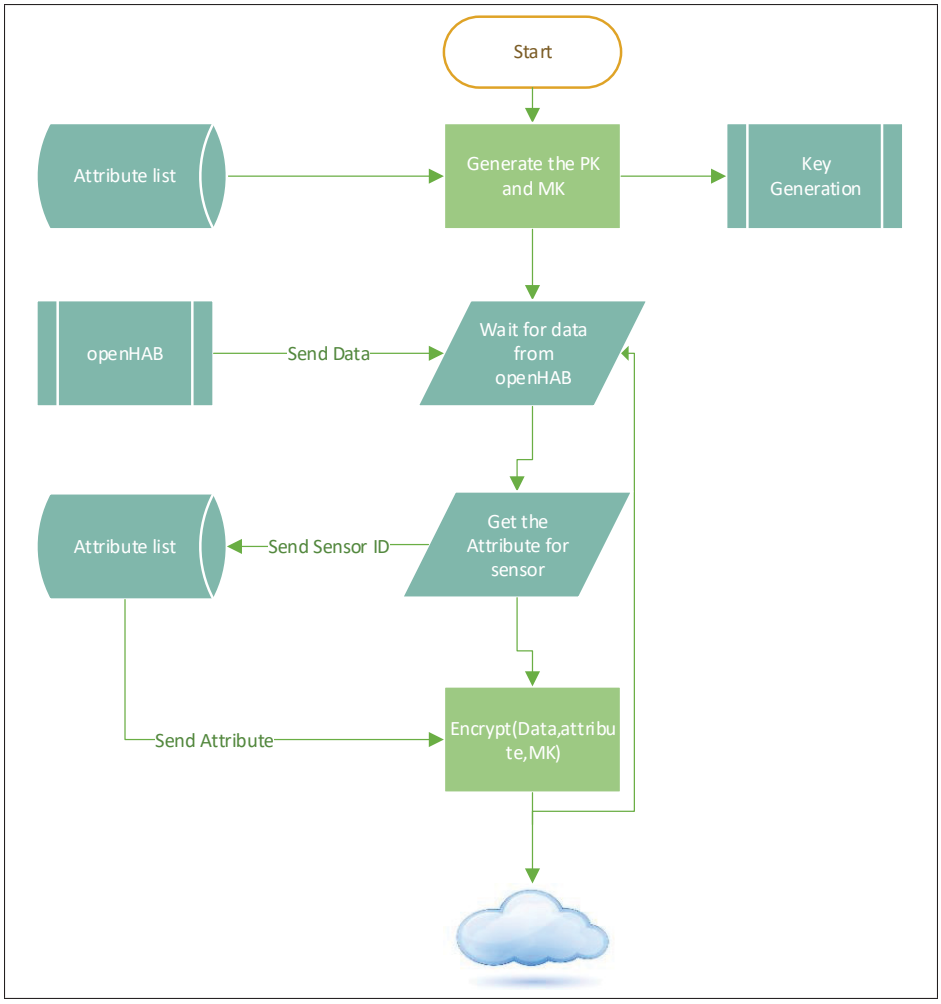


Figure 3.2 Encryption Module Flow Chart.

KeyGen Module (KG) is responsible for generating the user secret keys (SK) using the MSK which is generated from the EM and the privacy settings defined by the Home owner. The flow chart of the KG is shown in the Figure 3.3.

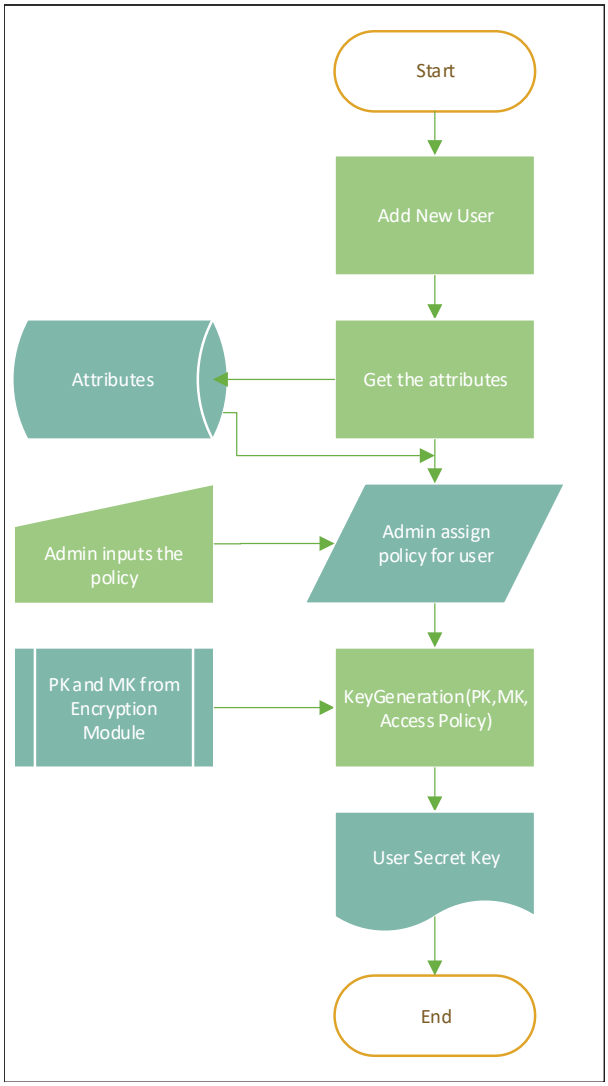


Figure 3.3 Key Generation Flow Chart.

Privacy Module (PM) provides the list of attributes required for defining the policy which is required by the EM and KG which is only accessible by the home owner.

Decryption Module (DM) is used by the service provider for decrypting the data using the unique SK. The flow chart of the DM is shown in the Figure 3.4.

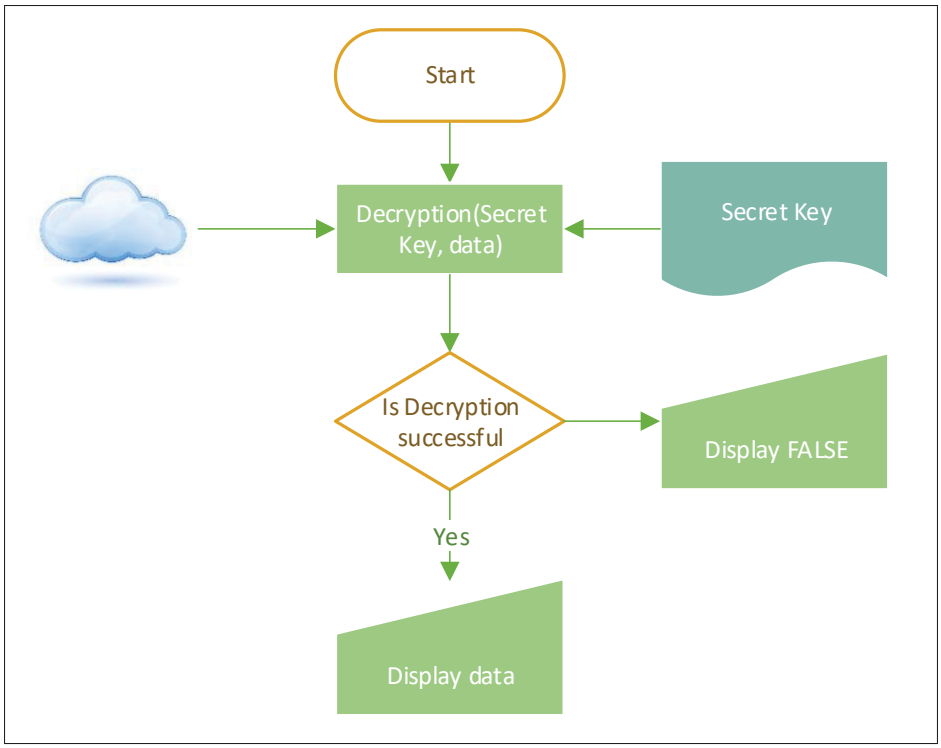


Figure 3.4 Decryption Flow Chart.

3.3 Implementation

In order to implement the privacy preserving architecture, we have used BeagleBone Black as the hardware hosting the middleware and the privacy solution as it has the lowest hardware configuration. The hardware and the software specification are illustrated in Table 3.1. To conduct our experiments, we adopted three ABE schemes: KP-ABE, CP-ABE and an enhanced version of KP-ABE (YCT) (Yao *et al.* (2015)). The details of the YCT scheme is shown in the Appendix I.

Table 3.1 Hardware and Software Specification

Processor	M335X 1GHz ARM Cortex-A8
RAM	512MB DDR3
Storage	2GB eMMC flash Storage
Operating System	Ubuntu Minimal
Program	OpenHAB

The EM and KG modules are implemented using python while PM is implemented using JavaScript. Using the openHAB's rules scripting, we defined the pipe-lining that allows the transfer of the sensor data collected to the EM module. We are ensuring the confidentiality of data throughout the system by using cross application data transfer. Then the data is encrypted before it is uploaded to the cloud. To define the privacy settings, we have used a form which serves as a user interface for the PM module, which interacts with the home owner (admin) allowing him to specify the access policies and attributes. In addition, the PM module interacts with openHAB to get the list of sensors, their types and their organization inside the smart home. Using the PM interface, the home owner can set up the parameters required for the EM and KG. Furthermore, the admin is responsible for adding users and third-party services, as well as assigning attributes to sensors. The KG module is responsible for generating cryptographic keys and transmitting them to the users and service providers. The generated keys satisfy the policies defined by the home owner and are updated periodically depending on the owner requirements. The service provider can get the data from the cloud and decrypt them using the decryption module running in their system, provided they have the required keys.

3.4 Test Scenario

In our test scenario, we have used three services requiring access to the different sensors available in the smart home (Figure 3.5). Service S1 and Service S2 are managed by two different residents of the house while Service S3 is managed by a utility company. As shown in Figure 3.5, we are simulating a house consisting of a living room, two bedrooms, a kitchen, a bathroom, and a utility room. The types of data generated by the various sensors are presented in Table 3.2.

Figure 3.6 shows the GUI provided by the PM module to set up the access policies. Here, we have shown the settings when the KP-ABE scheme is applied. The GUI allows assigning attributes to sensors and specifying policies overs attributes for services. The PM module provides also a dual GUI for the CP-ABE settings where policies are specified for sensors and attribute sets are assigned to services. The attributes set used for in our case is: A1, A2, A3,

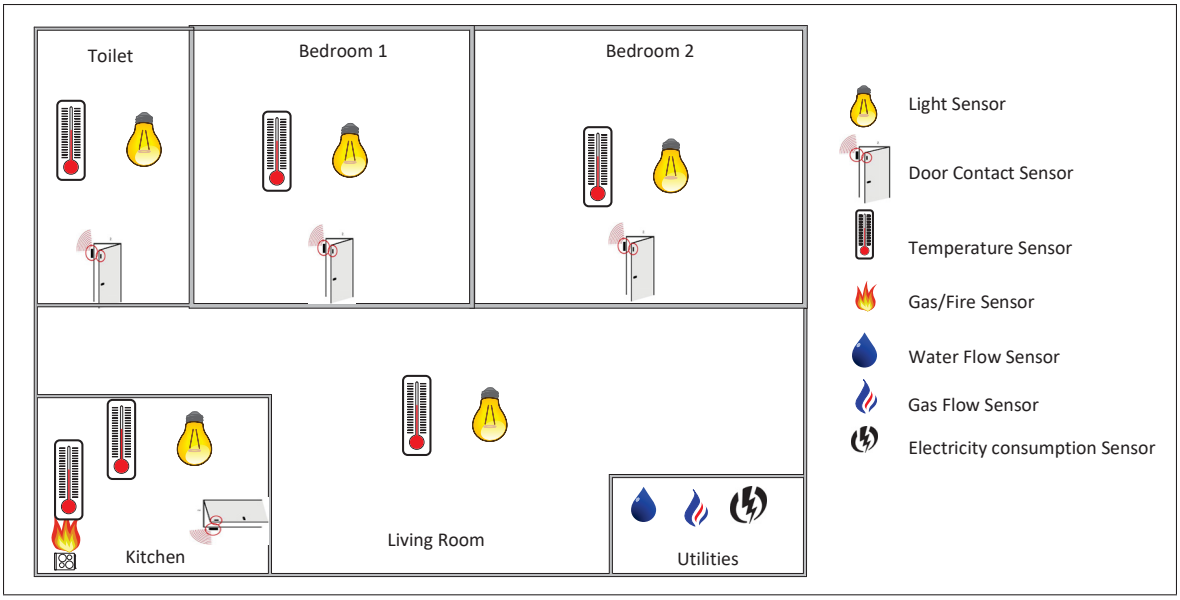


Figure 3.5 Test Schematics and Sensors Locations.

Table 3.2 Sensor Details and Locations

Sensor Type	**Locations**	**Data Type**
Light	Bedroom 1, Bedroom 2, living room, toilet, kitchen	String [dim, bright, very bright]
Temperature	Bedroom 1, Bedroom 2, living room, toilet, kitchen	Float [2 decimal places]
Contact [Door]	Bedroom 1, Bedroom 2, living room, toilet, kitchen	Binary [Open/Close]
Gas Dectection	Kitchen	String [low, medium, heavy]
Water Flow	Utility room	Integer
Electricity Consumption	Utility room	Integer
Gas Flow	Utility room	Integer

A4, A5, A6. An example of possible KP-ABE settings is presented in Table 3.3 (assigning attributes to sensors) and Table 3.4 (policies specification for the involved services).

The services use a web interface to get the data for all the sensors but they can only view the sensed data for which they have access right. For those sensed data for which a service does not have the right to access, the message ‘denied’ is displayed. From Figure 3.7 , we can see

Table 3.3 Services Access and policy

Service	Sensors Access	Access Policy
S1	Bedroom 1, living room, bathroom, kitchen	A1 or A2
S2	Living Room, Bedroom 2, Bathroom	A3 or A4
S3	Utility room	A5 or A6

Table 3.4 Attribute set for sensors

Sensor Location	Attribute set
Living room	A1,A2,A3,A4
Bedroom 1	A1,A2
Bedroom 2	A3,A4
Kitchen	A1,A2
Bathroom	A1,A2,A3,A4
Utility room	A5,A6

that Service S1 is denied from accessing the data collected from Bedroom 2 and Utility room. For the sake of demonstration, we added two other services.

- The 'Smart House' service granted access to all the sensors (holding all the required decryption keys). As shown in the first column of Figure 3.7, all the sensed data values are displayed in plaintext (successfully decrypted by the service).
- The 'Encrypted' service being denied access to all the sensors (having no valid decryption key). As shown in the second column of Figure 3.7, all the sensed data is displayed as ciphertext (unintelligible because the service failed to decrypt them).

3.5 Evaluation

In order to evaluate our implementation, we have measured the encryption overhead of three different ABE schemes: the CP-ABE and the KP-ABE implementations provided by (Charm) and an enhanced implementation of KP-ABE (Yao *et al.* (2015)). We have used all the nineteen

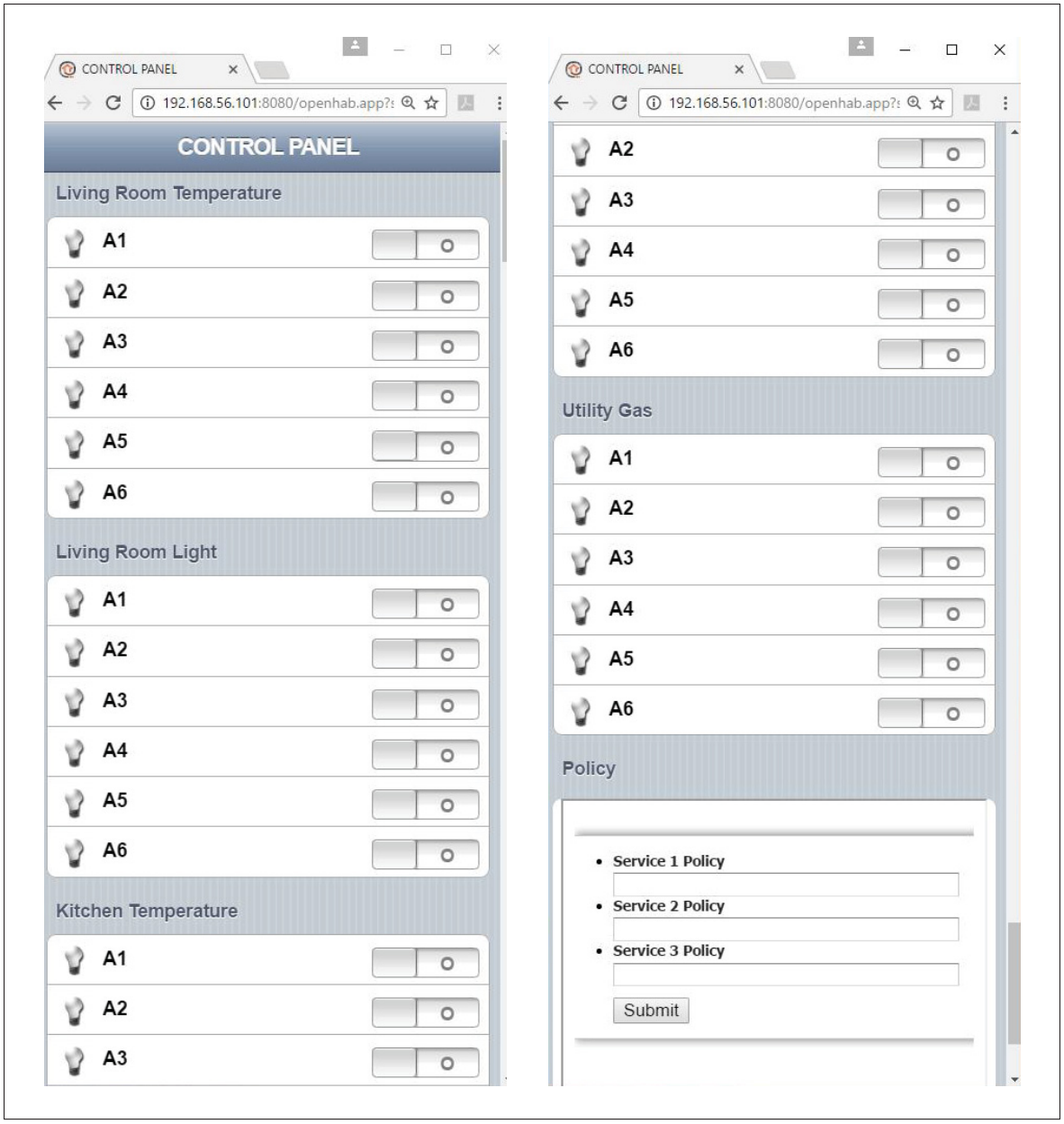


Figure 3.6 Assignment of Attribute Sets and Privacy Setting.

sensors presented in the test scenario. Each sensor sends periodically a new data value every twenty seconds. In addition, we have varied the number of used attributes or policy from five to thirty. We have measured the overall execution time required to encrypt the data received by the EM module from openHAB (Figure 3.8). More detailed results are presented in Figure

Figure 3.7 Screen Shot of Different Users.

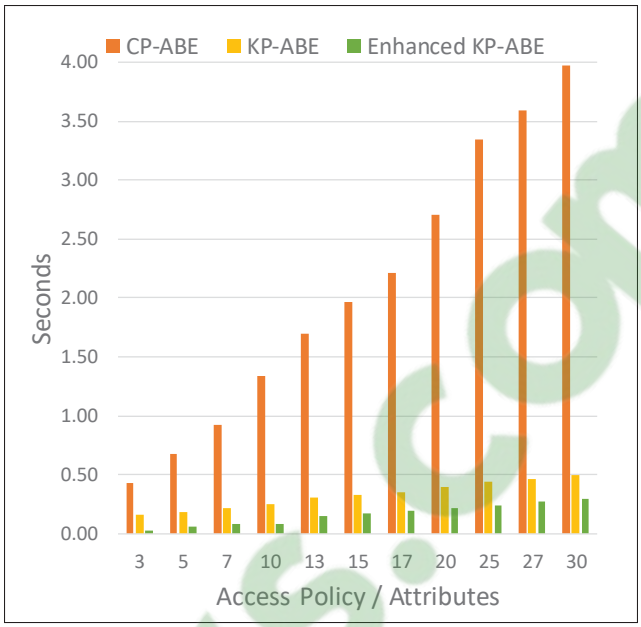
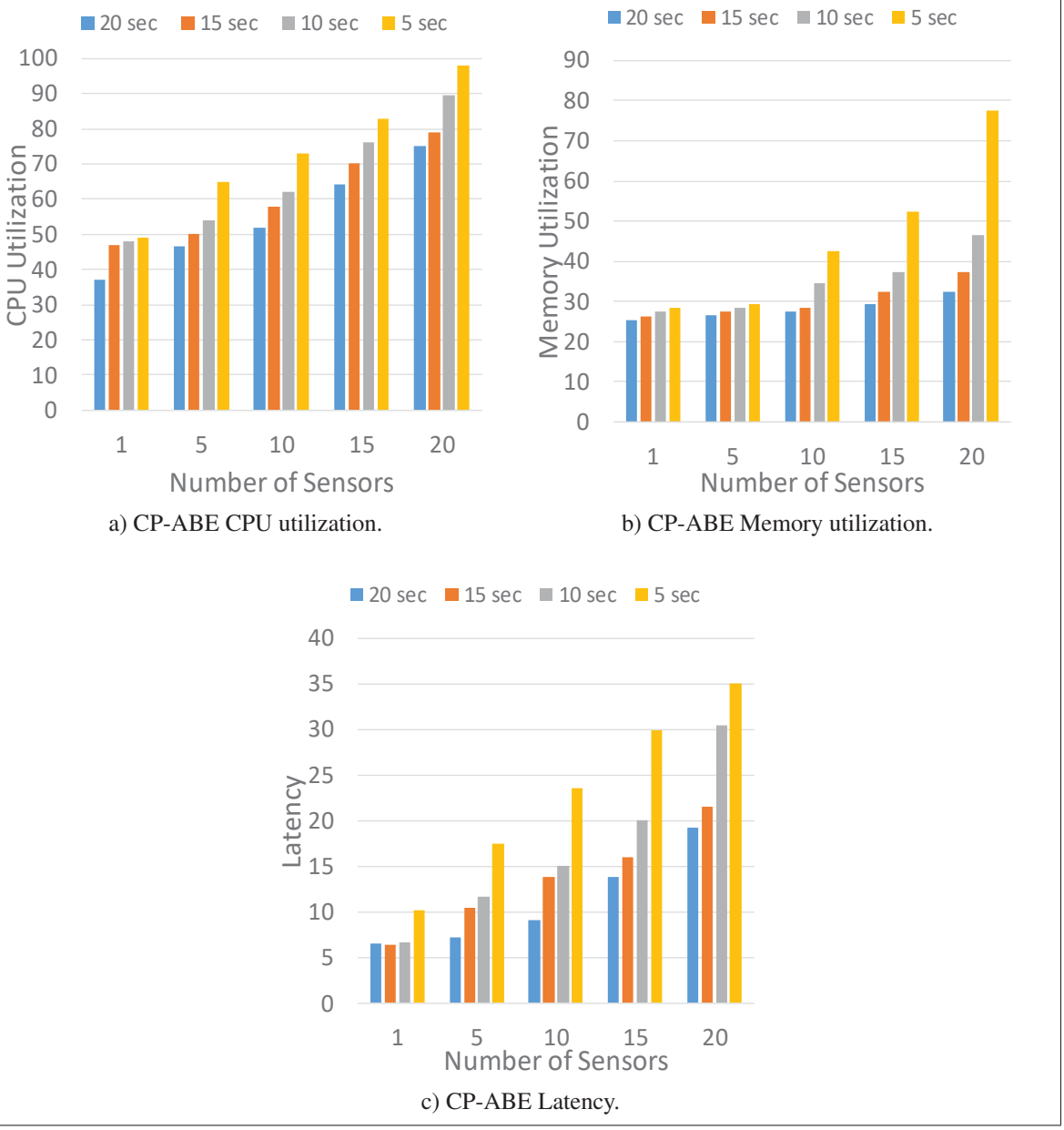


Figure 3.8 Time Overhead of ABE Encryption Schemes.

3.9a to Figure 3.11c, they depict the resources utilization and the latency of the overall process, in term of data frequency, number of sensors and keeping the attribute or policy fixed at thirty.

As shown in the experimental results, resources consumption and system latency increases gradually when the number of connected sensors and the frequency of the data are increased. However, more valuable conclusions can be extracted from careful investigation of the results shown by each figure. Let us start with the results related to CPU consumption. Depending on the maximum CPU% that can be granted to the system, some configurations become infeasible. For example, if the maximum CPU percentage is 50%, then the system equipped by CP-ABE cannot serve more than 5 sensors and the interval of collecting data for this maximum should be strictly greater than 5 seconds (Figure 3.9a). For the same CPU maximum percentage, KP-ABE is providing better capabilities (Figure 3.10a) being able to serve a maximum of 10 sensors provided that the interval of collecting data is greater than 10 seconds. For the third ABE algorithm (Figure 3.11a), a maximum of 15 sensors can be served if data is collected each 20 seconds. Based on the results related to memory overhead, almost all the configurations of the three algorithms are feasible when a maximum of 50% memory budget is available.



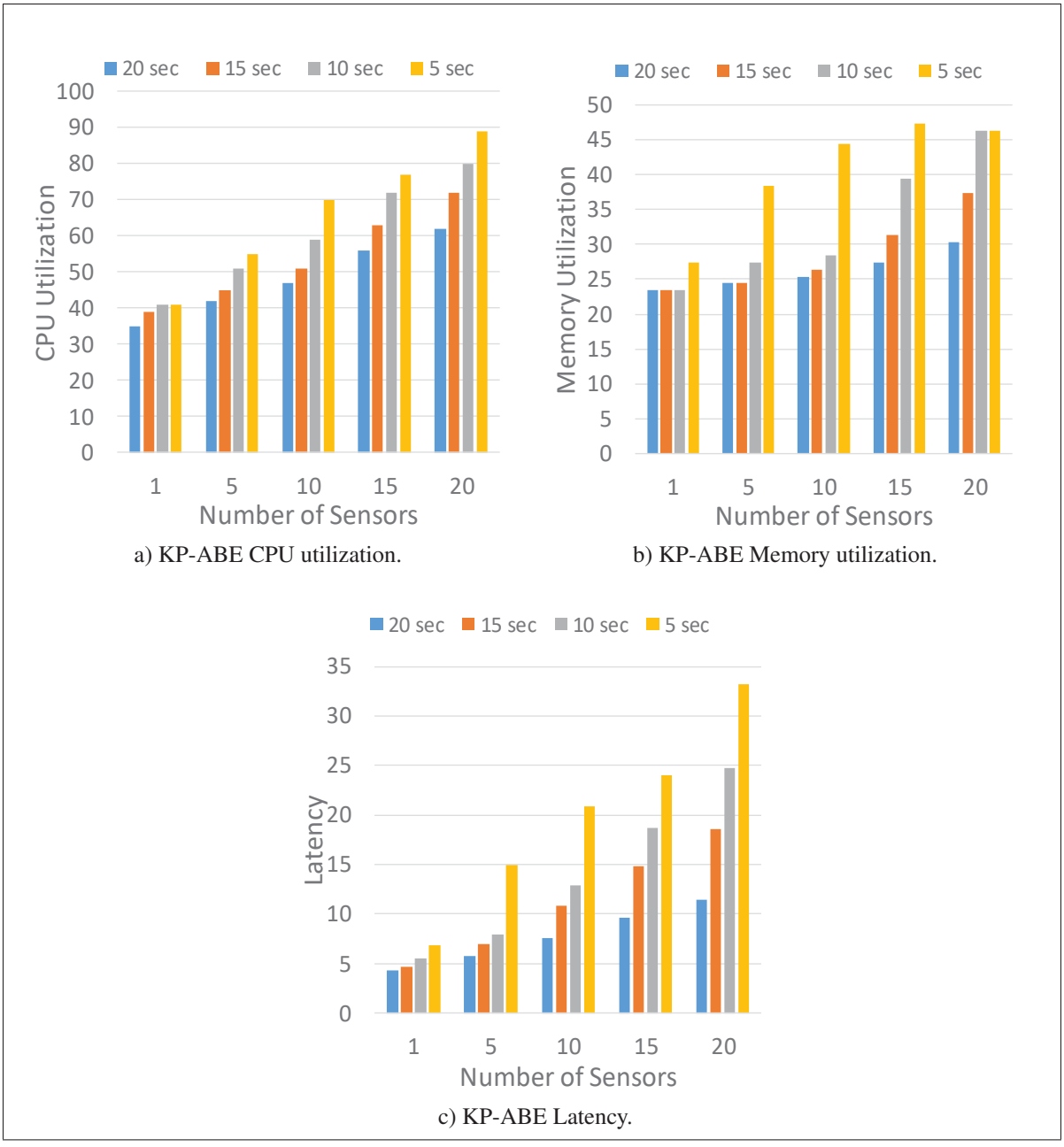
a) CP-ABE CPU utilization.

b) CP-ABE Memory utilization.

c) CP-ABE Latency.

Figure 3.9 CP-ABE results.

More precise conclusion can be extracted from the results provided by Figure 3.9b, Figure 3.10b and Figure 3.11b when less memory budget is dedicated to the system. The results related to latency are of paramount importance especially for environments where the most up to date data is required. In fact, if the remote services should be aware of any update within a



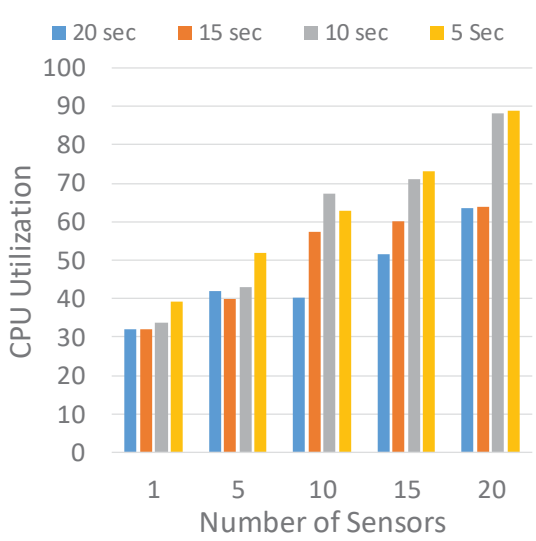
a) KP-ABE CPU utilization.

b) KP-ABE Memory utilization.

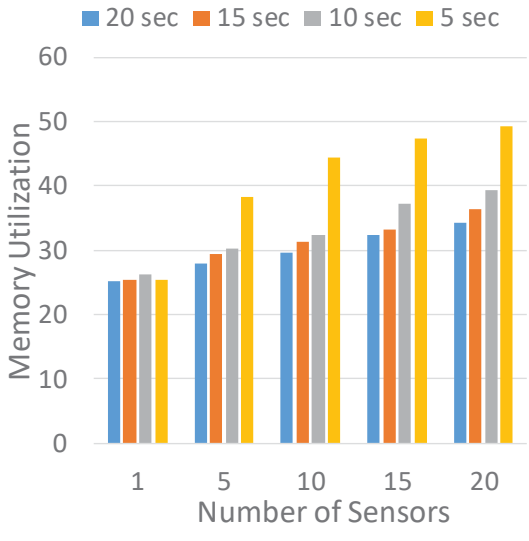
c) KP-ABE Latency.

Figure 3.10 KP-ABE results.

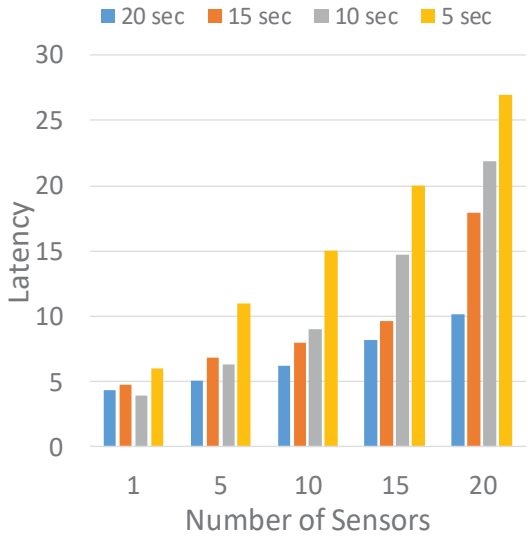
maximum of 5 seconds, then CP-ABE should be discarded. More precise conclusions can be extracted for other latency constraints.



a) YCT CPU utilization.



b) YCT Memory utilization.



c) YCT Latency.

Figure 3.11 YCT results.

3.6 Summary

In this chapter, we have proposed an architecture and incorporated ABE cryptographic schemes with openHAB, an well-known IoT smart home middleware. Then we have simulated a smart-home with multiple sensors, users with different access control and showed that our proposed

architecture is feasible with the middleware. Later we performed experimentations to show that the architecture is capable of using the different ABE schemes and illustrated the performance of the schemes in our architecture using resource utilization and latency of the system. From the experimental results, we saw that performance of CP-ABE is higher than other schemes and it is feasible enough to incorporate ABE in an smart home environment under certain condition based on number and frequency of the sensors.

CHAPTER 4

OUTSOURCING ENCRYPTION IN A SMART HOME

ABE is one of the most prominent way to achieve dynamic confidentiality, privacy and access control along with data encryption and integrity. The main drawback of ABE is that it requires more resources than other asymmetric encryption schemes as shown from the experimentation shown in the Chapter 3. Also since the IoT devices are built in way to achieve longevity and they are used mostly to serve a specific purpose, so they are designed to have less resources compared to the IoT edge devices as well as the cloud. So, in order to reduce the computational overheads we propose a framework that does partial encryption at the gateway and outsource most of the heavy computation to the cloud. However, we cannot trust the proxy server always, so we have used a solution that uses the concept of partial encryption at the gateway and rest of the computational processing at a proxy server running in the cloud.

4.1 Dummy Attribute ABE scheme

In this chapter, we are using the ABE offloading technique that is proposed by (Jin *et al.* (2015)). The scheme uses the concept of dummy attribute and uses a specific access tree as shown in Figure 4.1. The algorithms for this scheme are as follows:

Setup $\rightarrow$ **(MK, MSK)**: The algorithm generates the PK and MSK which is required by the encryption and key generation algorithms.

KeyGen (PK, MSK, ω**)**$\rightarrow$ SK: KeyGen (Key Generation) take the PK, MSK, and ω as input. ω is list of attribute of the user. The output of this algorithm is the secret key SK.

Encryption$_{Dummy}$ **(Message,** α**)**$\rightarrow$ **CT**$_{Dummy}$: At this algorithm the user encrypts his/her data with α where α is the access policy. The algorithm calculates the shares of the of the access tree and encrypts the message with the dummy attribute of the access tree. The output of this algorithm is the CT$_{Dummy}$.

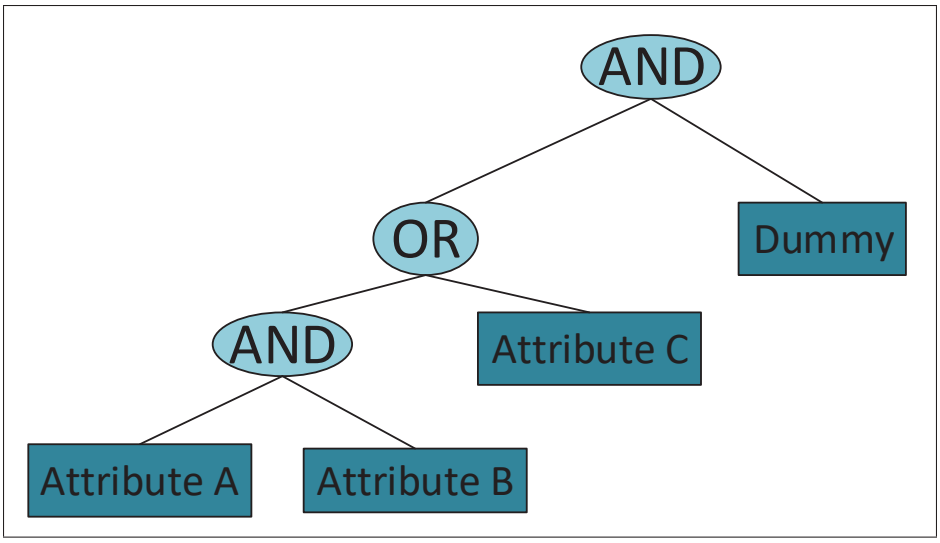


Figure 4.1 Dummy Access Tree.

Encryption$_{Actual}$ **(CT**$_{Dummy}$**, PK)**$\rightarrow$ **CT**$_{Actual}$: At this stage the algorithm uses CT$_{Dummy}$ and calculates the complicated exponential polynomial calculations on each shares and generates the actual ciphertext CT$_{Actual}$ which includes the CT$_{Dummy}$.

Decryption (SK, CT$_{Actual}$**)**$\rightarrow$ **Message**: This algorithm is used by the clients who use their secret key to recover the message.

4.2 Architecture

Figure 4.2 shows our architecture on smart home ecosystem. The architecture divides into five modules as the following:

Data Collector Module (DCM) is used for receiving data from the sensors. DCM acts as a middleware in the smart home gateway. The module run threads for accepting connection from the sensors, then it verifies and authenticate the sensors' information and prepares the data into a specific format then forwards to Gateway Encryption module of the home gateway.

Gateway Encryption Module (GEM) is responsible for encrypting the data coming from the DCM. When a data receives from the DCM it checks the database and finds the policy

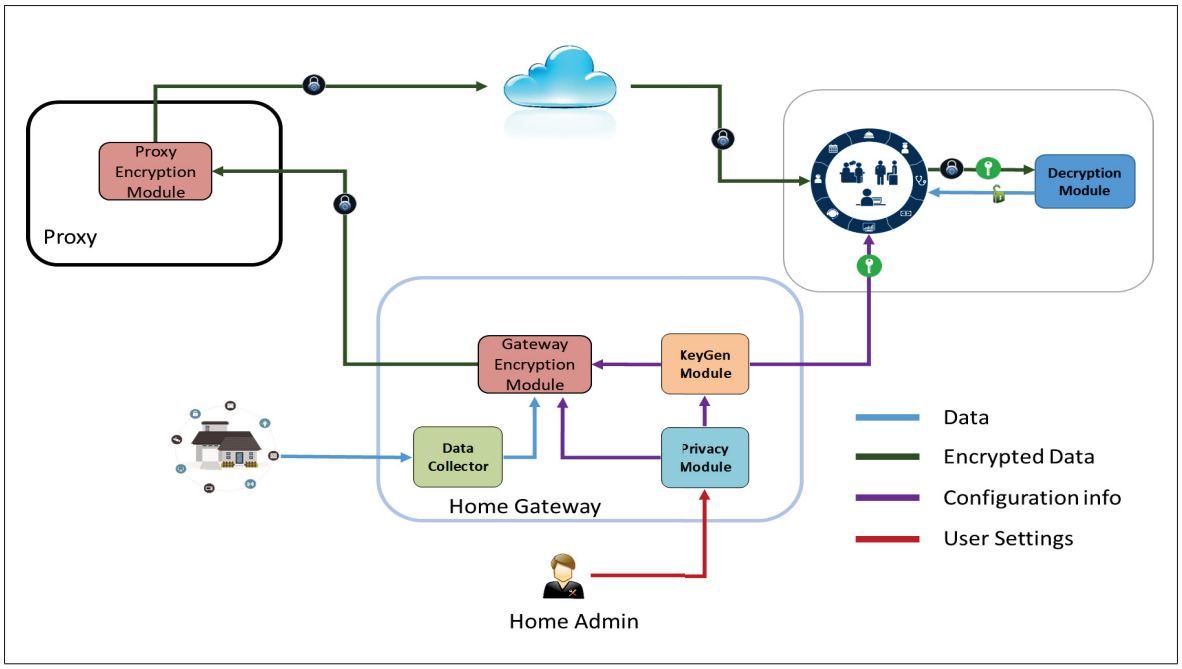


Figure 4.2 Outsourcing Architecture Overview.

corresponding to that sensor including the dummy attribute. Then the module encrypts it with the dummy attribute, calculates the shares and sends it to the Remote Encryption Module.

Proxy Encryption Module (PEM) is responsible for encrypting the data from GEM using the actual policy/attributes which is set by the owner. PEM checks the message for the shares and uses that shares to perform the polynomial computation to prepare the final CT of data and sends to the cloud for storage.

KeyGen Module (KM) is responsible for generating the PK and MK which are required by the GEM for encrypting the data and SKs for the services for decrypting. During generating the SK the KM incorporates the dummy attributes as well.

Privacy Module (PM) interacts with the admin for setting up the primitives required by the GEM and KM which is the incorporating a unique dummy attribute to each policy and the attribute set of the users and as well as policy for the data encryption.

Decryption Module (DM) decrypts the CT using the SK. If the secret key satisfies the access policy then the data is displayed else "Denied" is diplayed.

4.3 Implementation

The implementation of the modules are implemented using python and also we are using charm (Akinyele *et al.* (2013)) (Charm) for their crypto modules. The data from the sensors are converted to bytes before it is feed into the GEM. We also used the serialize and de-serialize functions of charm for the conversion of cipher text into byte codes, so that the parameters and the message become unreadable even more. In order to evaluate the performance of our framework we are using a custom data collector (DCM) which serves as a middleware. Our framework can used as plug and play with any other smart-home middlewares. In order to keep the environment simple, we are using TCP for data communication. We will be simulating a smart home with different types of sensors places in different locations of a house. All the sensors will send data to the gateway, which is in our case, is Raspberry Pi which will act as the DCM and GEM, and a desktop PC which will act as PEM. The configurations of the gateway and desktop is displayed in table 4.1 and 4.2. Table 4.3 shows the type of data generated by the sensors and their identification.

Table 4.1 Hardware and Software Specification of Gateway

Processor	1.2GHz 64 bit quad-core ARMv8
RAM	1GB
Storage	16GB eMMC flash Storage
Operating System	Raspberian Debian OS

4.4 Test Scenario

We will be evaluating our architecture with different types of ABE using the same scenario illustrated in Figure 3.5. We are using one dummy attribute for GEM which is unique attribute assigned to each sensors and based on access we will assign those dummy attributes for the key

Table 4.2 Hardware and Software Specification of Proxy

Processor	Intel i7 3.2 GHz Quad Core
RAM	16GB RAM
Storage	320GB
Operating System	Ubuntu

generation. In addition, we will have three services which will have access to the sensor data based on the ABE policy. The configuration for CP-ABE is shown in table 4.5 and in table 4.4 is for KP-ABE.

Table 4.3 Sensor Information and data type

Sensor type	**Sensor ID**	**Data type**
Light	Light1....Light8	String [Dim, Bright, Very Bright]
Temperature	Temp1....Temp7	Float [2 decimal places]
Contact [Door]	Cont1....Cont8	String [Open/close]
Smoke Detection	Smoke1....Smoke4	String [low, medium, heavy]
Water Flow	Water	Integer
Electricity consumption	Elec	Integer
Gas Flow	Gas	Integer

Table 4.4 KP-ABE settings

Sensor ID	**Dummy Attribute**	**Actual Attribute**
Light1	D1	Attribute1 .... Attribute4
Temp1	D9	Attribute5 .... Attribute8
Cont1	D16	Attribute9 .... Attribute12
Smoke1	D24	Attribute13 .... Attribute16
Water	D28	Attribute17 .... Attribute20
Elec	D29	Attribute21 .... Attribute24
Gas	D30	Attribute25 .... Attribute28

Home admin or the owner is responsible for setting up the attributes and policies required for the KM and PEM by using the PM. PM is a graphical user interface as shown in Figure 4.3 for KP-ABE setup where the home admin can select the attributes for the sensors and write policies

Table 4.5 CP-ABE settings

Sensor ID	Dummy Policy	Actual Policy
Light1	D1	Attribute1 AND Attribute2 AND ... Attribute4
Temp1	D9	Attribute5 AND Attribute6 AND ... Attribute8
Cont1	D16	Attribute9 AND Attribute10 AND ... Attribute12
Smoke1	D24	Attribute13 AND Attribute14 AND ... Attribute16
Water	D28	Attribute17 AND Attribute18 AND ... Attribute20
Elec	D29	Attribute21 AND Attribute22 AND ... Attribute24
Gas	D30	Attribute25 AND Attribute26 AND ... Attribute28

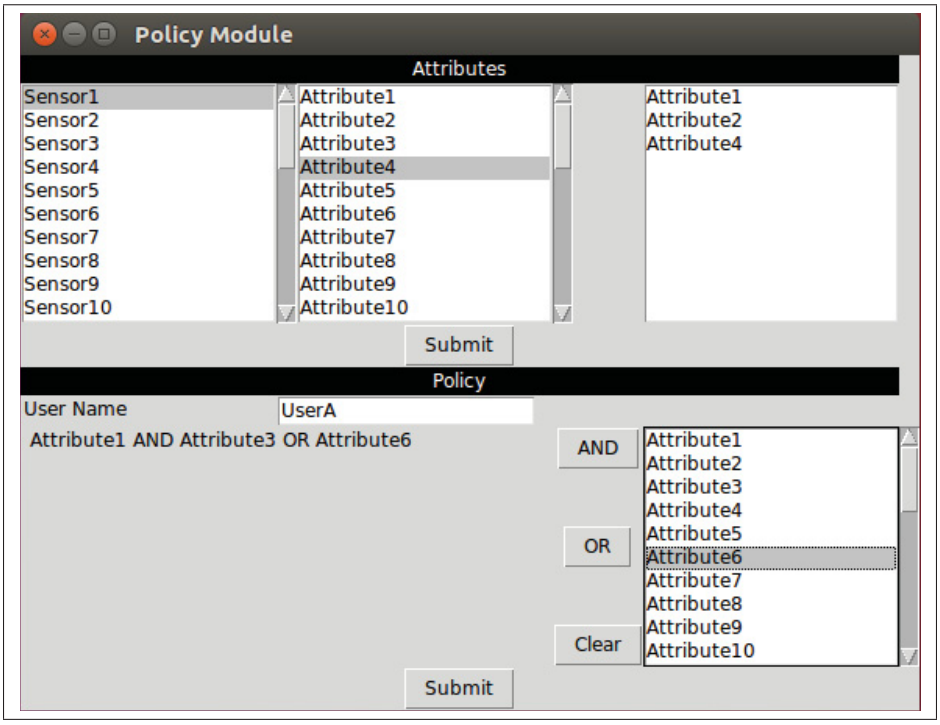


Figure 4.3 Policy Module.

for the services for generating the secret key. For CP-ABE the interface is reversed; attributes for services and policy for the sensors. KG is responsible for setting up the environment by generating the PK and MK, and then transferring them to the GEM. KM is also responsible for generating the secret keys of the services and transmitting to the designated service in a secured manner. DCM acts a smart home middleware and collects data from the sensors and forwarding to the GEM using inter processing communication. GEM is responsible for encrypting the data

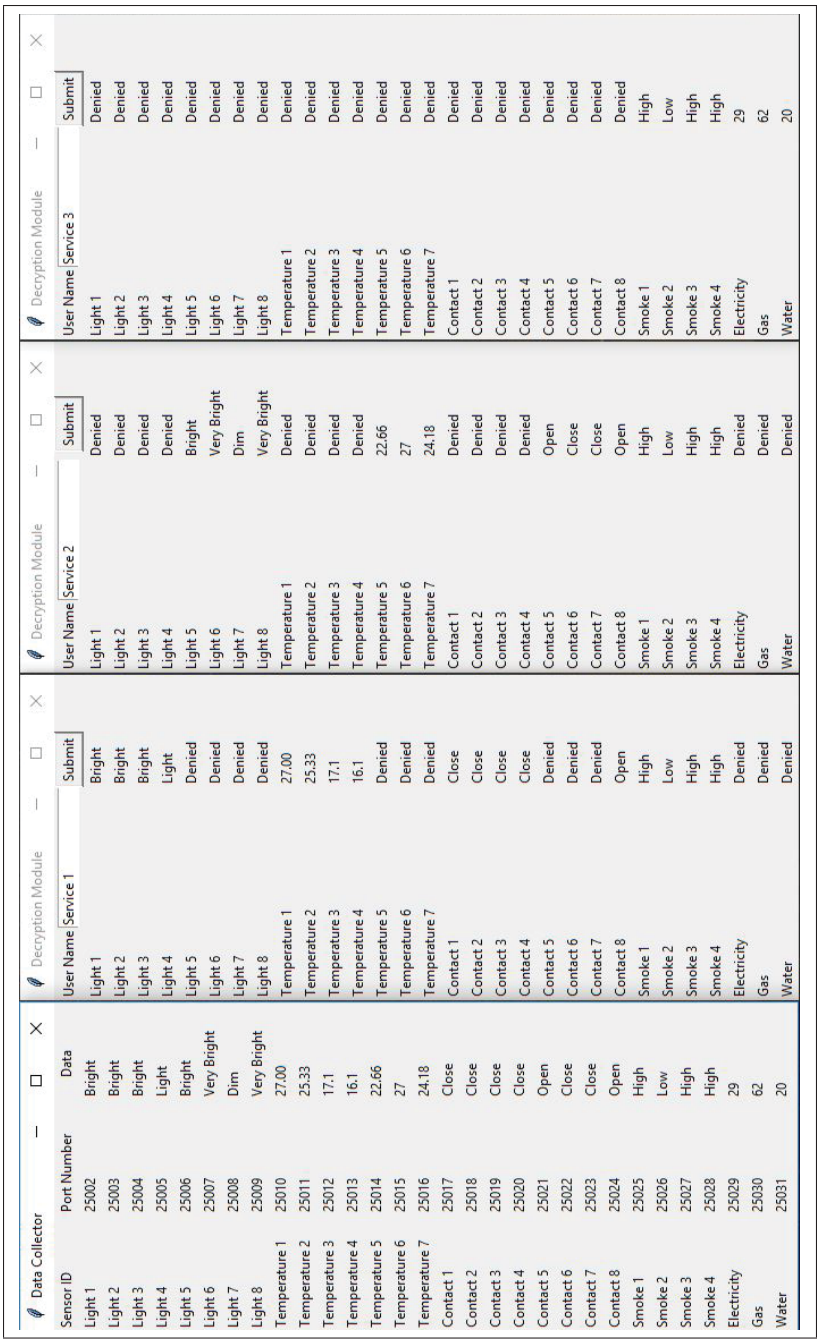
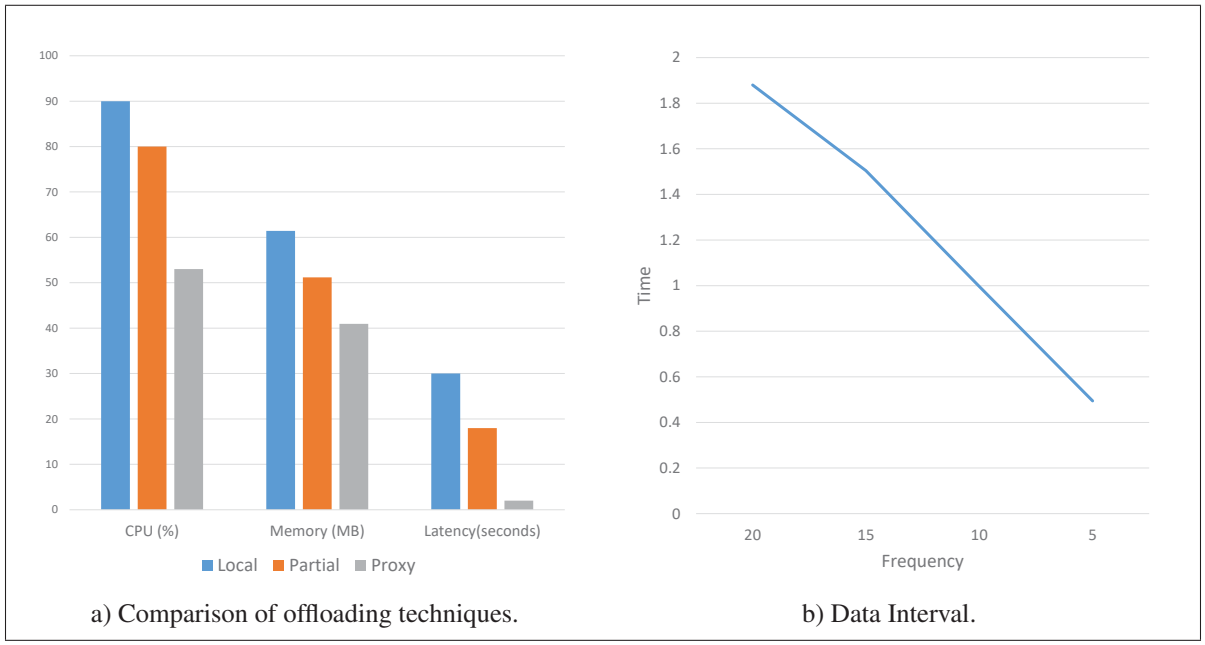


Figure 4.4 Screen shot of data collector and different service.

using one dummy attribute and then the data is being transferred to the PEM. PEM does the encryption with actual attributes/policy which is set by the home admin and then the encrypted data is stored in the cloud. The services can only decrypt the data from the cloud if they have the

decryption module running in their system and has a valid key, which satisfies the requirement of ABE. Figure 4.4 shows the view of different modules and services. From the left of Figure 4.4 is the visual representation of DCM and the rest are different services who have access to different sensors. If the service have access to that sensor they can view the value else denied message is shown based on the access policy of the services.

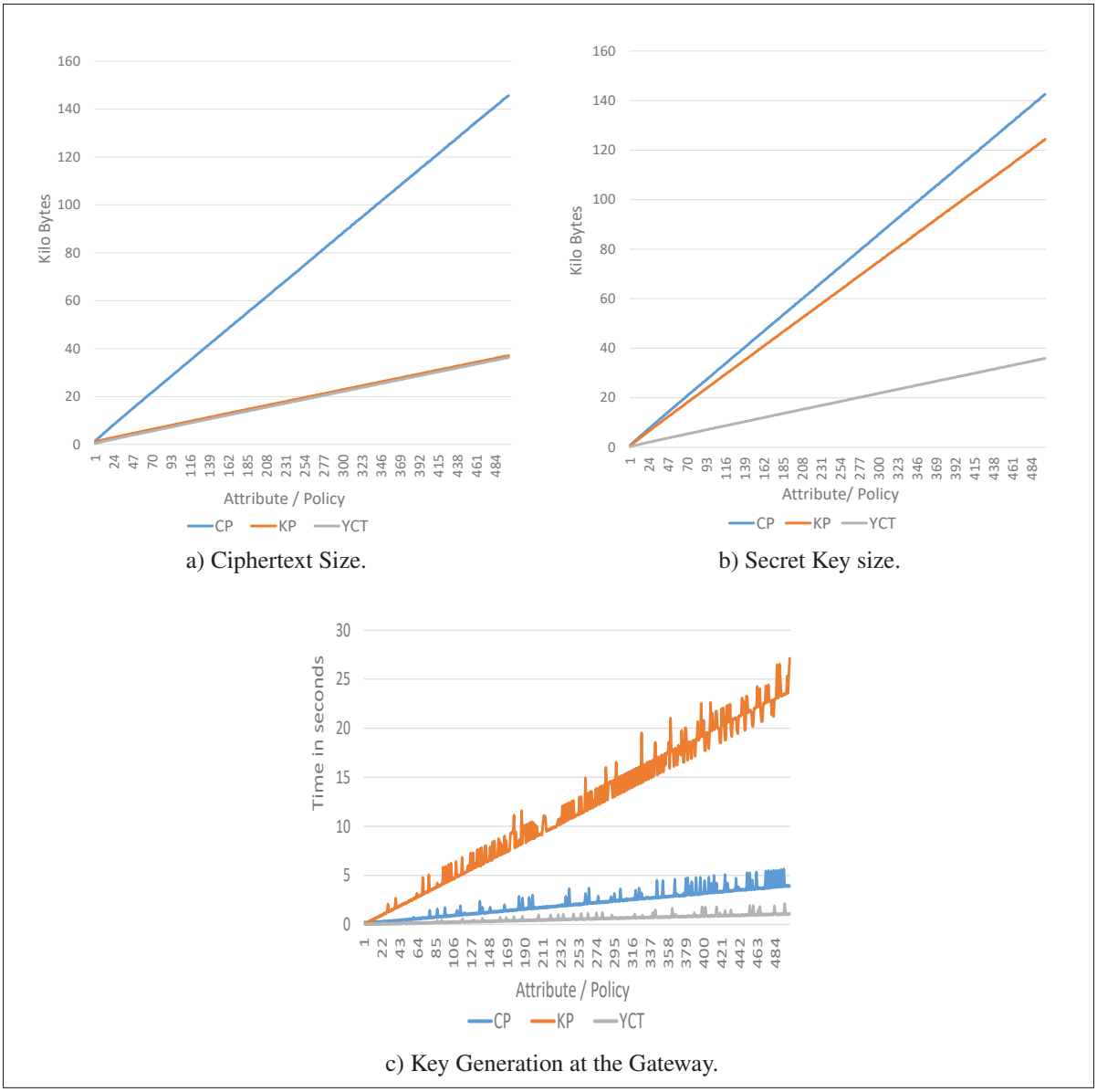
4.5 Evaluation



a) Comparison of offloading techniques. b) Data Interval.

Figure 4.5 Experimentation on different techniques and settings.

In order to evaluate our framework, we will be performing experimentation based on the CPU, Memory, Power consumption and Latency of the KP-ABE, CP-ABE and a revised version of KP-ABE. In the experimentation we will be using length of 30 attributes/policy, 30 sensors and sample of 150 data at maximum for the ABE settings. We test different KP-ABE and CP-ABE schemes on our architecture to check which scheme are more suitable in smart home scenario. In our experimentation we are using a python script to determine the resource utilization, to calculate the latency we measure the difference between the time when the data enters the



a) Ciphertext Size.

b) Secret Key size.

c) Key Generation at the Gateway.

Figure 4.6 Initial results.

DC and when the encryption is fully completed. Also an USB tester to calculate the power consumption of the Raspberry Pi.

In Figure 4.5a shows the performance of the resource utilization (CPU, Memory and Latency) of gateway of CP-ABE using offloading the encryption. In this experimentation, we performed the encryption at a local gateway (Raspberry pi), part of encryption at gateway and part at proxy, and then all the encryption at the proxy server. Figure 4.5b explains the frequency

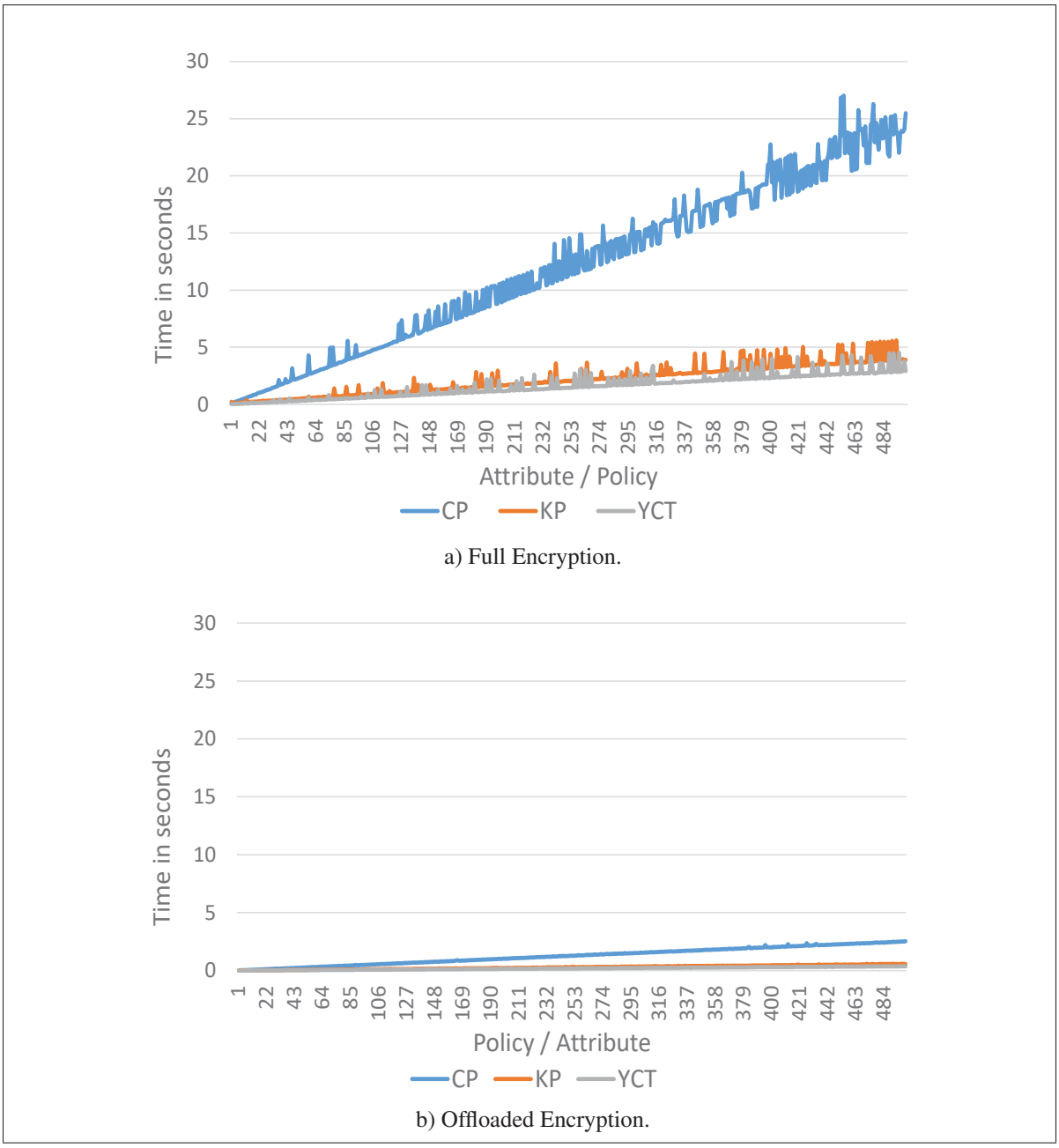
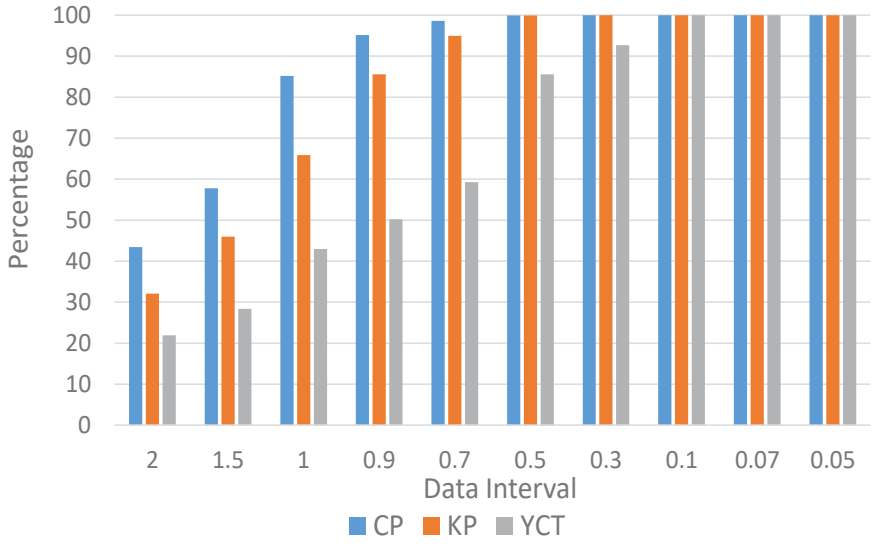
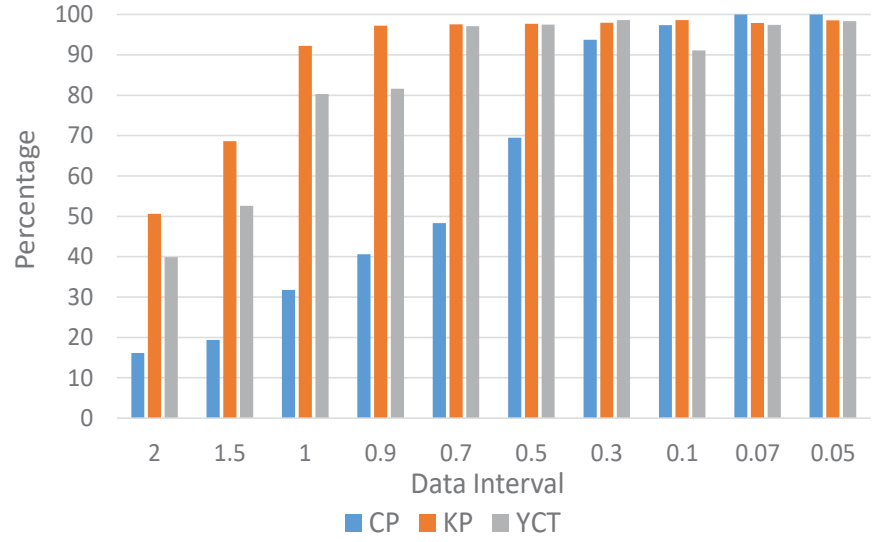


Figure 4.7 Encryption Overhead.

of data coming to the middleware based on the number of sensors and their rate of sending data. In this experimentation, we used 30 sensors and varying the sending rate of data from 20 seconds to 5 seconds. Figure 4.6a shows the size of cipher text using a constant message of 10 character with varying the attributes/policy. Figure 4.6b displays the secret key size with



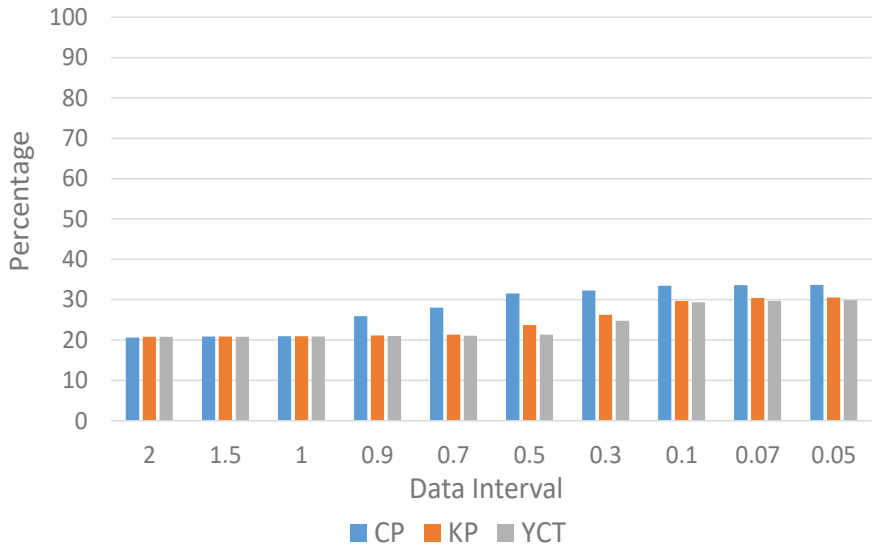
a) Full Encryption CPU Utilization.



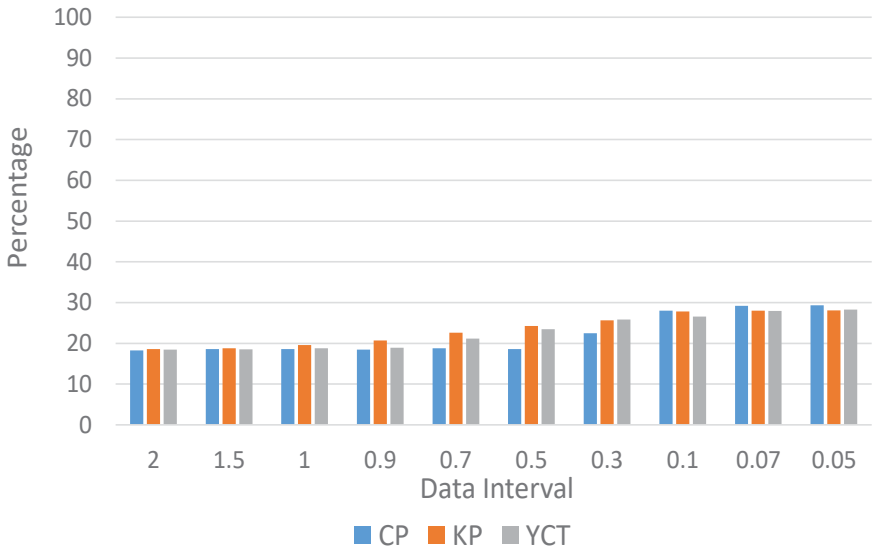
b) Offloaded Encryption CPU Utilization.

Figure 4.8 CPU Utilization.

different attributes/policy. Figure 4.6c shows the key generation varying the attributes from 1 to 500 at the gateway. Figure 4.7b and 4.7a displays the execution time of full encryption and offloaded encryption. Figure 4.8a, 4.8b, 4.9a and 4.9b shows the resource utilization for data interval 2 seconds to 0.05 seconds when sensors are introduced in the system for full and



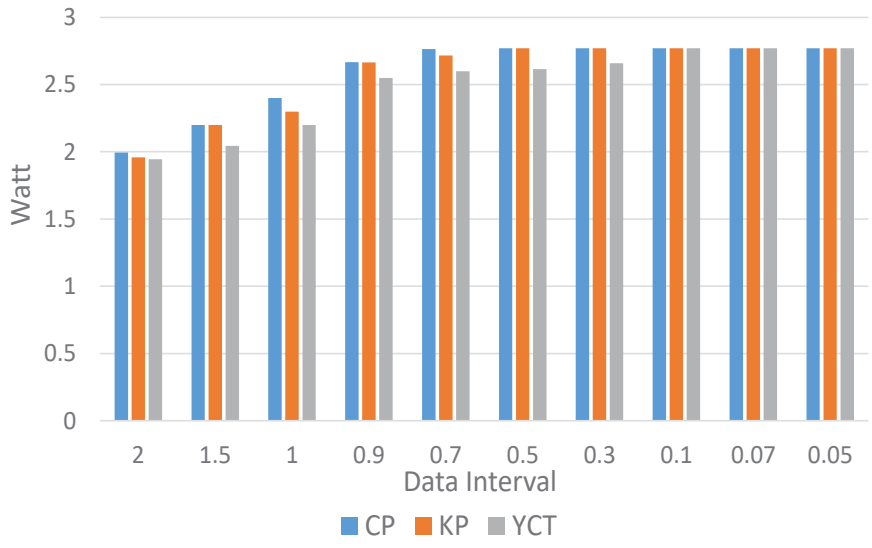
a) Full Encryption Memory Utilization.



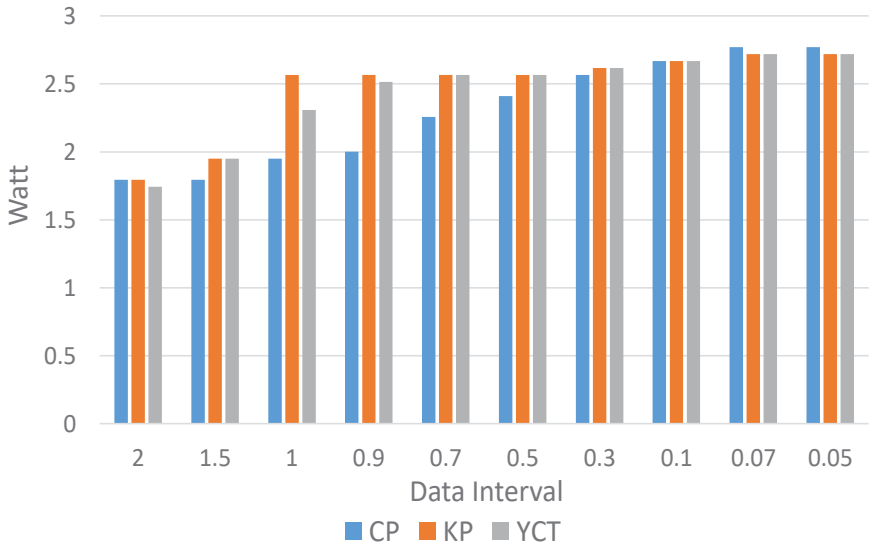
b) Offloaded Encryption Memory Utilization.

Figure 4.9 Memory Utilization.

offloaded encryption. Figure 4.10a and 4.10b displays the power utilization of the gateway with the interval of data. Figure 4.11a and 4.11b shows the latency of our system with respect with the data interval.



a) Full Encryption Power Consumption.



b) Offloaded Encryption Power Consumption.

Figure 4.10 Power Consumption.

From the results of Figure 4.5b, we see that the interval of data coming increases from 2 seconds to 0.5 seconds when the frequency increased from 20 seconds to 5 seconds. In Figure 4.5a we can see that if the system performs all the computation at the remote server, the resource utilization and latency decreased by 40%. From the Figure 4.6c shows with the higher number

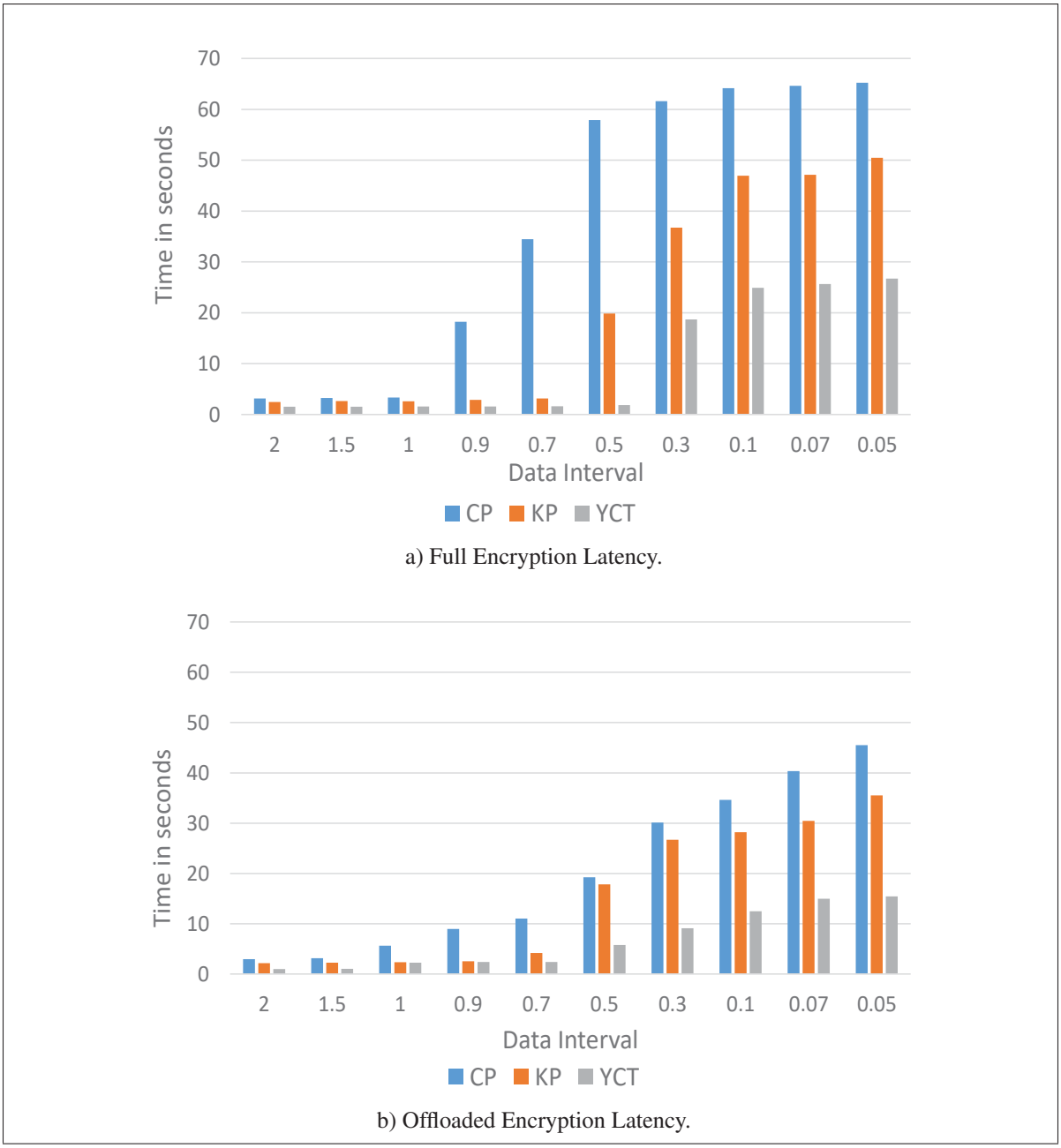


Figure 4.11 Latency.

of policy/ attributes the execution time of Secret Key Generation increases. From Figure 4.6b and 4.6a we can determine that with the increase of attribute/ policy the size of the message also increases and CP-ABE has the highest in size compared to YCT and KP-ABE. If we perform all encryption at the Gateway (see Figure 4.7a the execution time increases gradually for

the number of attribute where as if we do partial encryption at the gateway and offload the rest to the proxy (see Figure 4.7b)the execution time is decreased ten times the execution at the gateway and from these two figures, we can evaluate that the encryption have a very big impact where the execution is taking place. From the Figure 4.8a and 4.8b, we can see that the cpu utilization of CP-ABE 20% higher than other ABE schemes at data interval 2 to 0.5 and then gradually increases in the GEM for both cases. Also we can notice that the CPU consumption is lower when we do partial encryption rather than full encryption. In Figure 4.9b and 4.9a the memory utilization is almost similar for all the schemes and they are below 40% of the total memory available. From Figure 4.10b and 4.10a we can say that the power consumption does not go above 2.7 watt when the data interval is at 0.05 seconds. Further evaluation can be drawn from Figure 4.11a and 4.11b which shows the latency of a data which it requires for the whole process. We can see that the CP-ABE has the highest latency among the schemes and YCT has the lowest for both cases. If the system have a threshold of 10 second latency then:

- For full encryption minimum data interval for:
  - CP-ABE is 1 second;
  - KP-ABE is 0.7 second;
  - YCT-ABE is 0.5 second.
- For partial encryption minimum data interval for:
  - CP-ABE is 0.7 second;
  - KP-ABE is 0.7 second;
  - YCT-ABE is 0.3 second.

In conclusion, doing partial encryption at the gateway and offloading the rest to the proxy reduces the resource consumption and mainly latency of the data by 30% than doing full encryption process in the gateway.

4.6 Summary

In this chapter, we have shown how we can offload the encryption process to a proxy server even though the proxy is not trusted for computation, where we partially encrypt the data at the gateway and rest of the computation is being offloaded to the proxy server. We have presented the architecture and we have simulated a smart environment that our proposed architecture is feasible. Later we have experimented with different ABE schemes based on the resource consumption and latency based on interval that the gateway is receiving data.

CHAPTER 5

FUTURE WORK

During the last few years, the computation offloading from resource constrained devices are becoming popular. (Khan (2015)) did a comprehensive survey of the differnt offloading strategies to improve the performance of the mobile applications. (Tout *et al.* (2017)) proposed an intelligent model for computation offloading of mobile devices using a central decision engine which decides whether the data will be processed at the device or it should be offloaded to the cloud based on the resources available. (Tripathi (2017)) surveyed the issues regarding the offloading techniques and also provided some solutions using adaptive computation offloading for the MCC. (Cao & Cai (2018)) proposed a multi-user fully distributed computation offloading to Cloudlets using a theoristic machine learning approach. (Sharma *et al.* (2017)) (Kovachev *et al.* (2012)) (Chen *et al.* (2016)) (Kemp *et al.* (2010)) proposed different offloading techniques for adaptive computational offloading of MCC. (Shukla & Munir (2016)) proposed a computation offloading scheme where the IoT devices requests resourceful devices to perform the computation of its behalf. (Mazza *et al.* (2016)) used a cluster based computation offoading to reduce the energy consumption and execution of smart mobile devices in smart cities. (Samie *et al.* (2016)) (Wang *et al.* (2017)) (Kattepur *et al.* (2016)) also showed different ways to offload computation from resource constrained devices to the cloud.

As shown in the dissertation, that one of the best way to achieve access control, security and integrity requirement according to the new regulations and users' privacy is to use Attribute-Based Encryption (ABE) which provides access control as well as data encryption. But ABE requires more resources than other symmetric or asymmetric encryption schemes. There are lot of implementation of ABE to cope with the resource constrained devices but the latency of those implementations are still too high and also in most cases, the IoT devices in an smart environment remains idle during most of the time. So we are proposing a way of using load-balancing the encryption using idle devices in a smart-home as well IoT edge devices before the data is stored in the cloud as our future work which is shown in Figure 5.1.

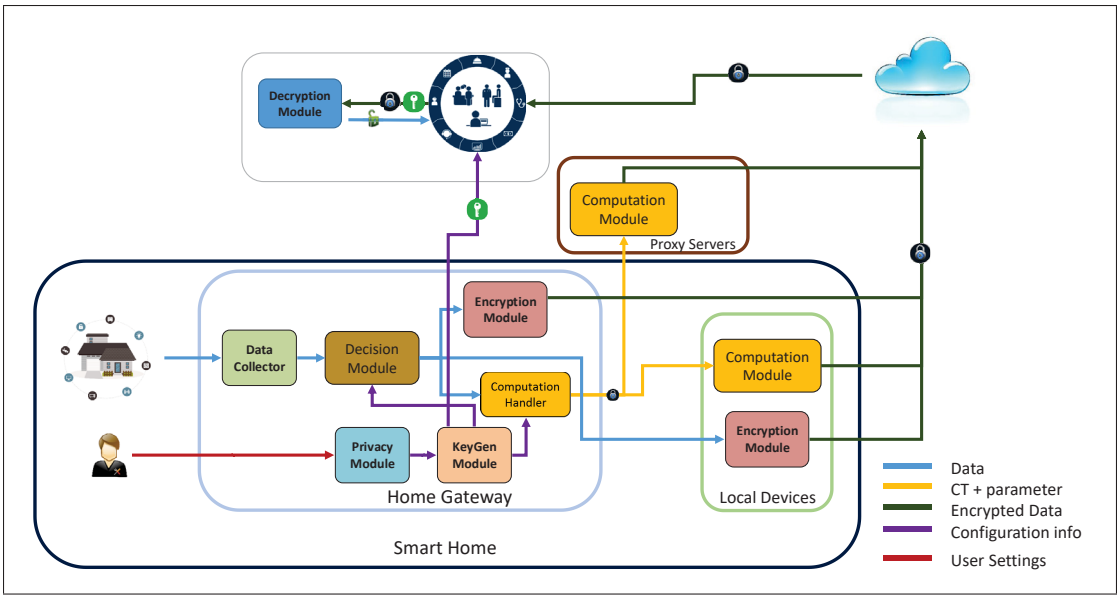


Figure 5.1 Load Balancing Architecture.

Data Collector is the interface where the sensors in the smart home communicate with each other. **Decision Module** decides what type of encryption and where the encryption will be done. **Privacy Module** interface for setting the policy and access control of the home owner and service provider. **Encryption Module** will do the encryption based on policy of the sensor. **KeyGen Module** is responsible for generating the keys for the encryptions and the service provider. **Computation handler** will perform partial encryption and package for the full computation of the encryption for the computation module. **Computation Module** is responsible of computation of the partially encrypted data into a complete cipher text.

The decision module will decide the best selection for encryption based on the length of the policy, data type and available resources available to have the lowest latency with least resource consumption along with the highest privacy level. Classification of choice is shown in the Figure 5.2.

Based on the input, it will either choose full encryption at the gateway or other idle devices in the smart-home or it will choose partial encryption on the idle local devices or the proxy servers. The decision module will be have the priority of doing full encryption at the high-

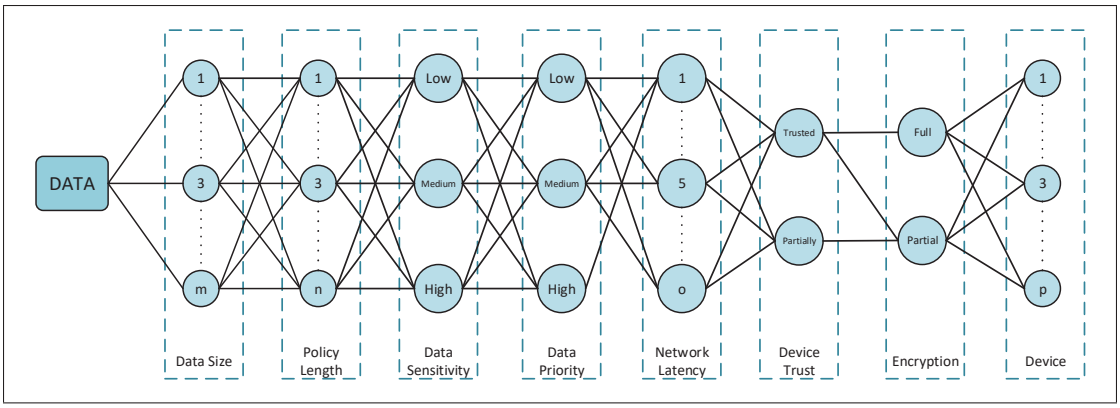


Figure 5.2 Classification rule.

est level or partial at the medium level based on the settings of sensors' data priority. As the resource, execution time and latency depends on the number of policy, frequency of data and the hardware specification of the devices as well as the trust of the device will effect the decision of the algorithm. The objective of the approach is to minimize resource consumption and execution time of the encryption process as well as to decrease the latency of the overall process.

In summary, in this chapter we have shown how we can extend the research to utilize the idle devices in an smart environment by using Machine Learning approach for the encryption. We have also presented a preliminary architectural model, classification overview and classification rules for the encryption load balancing technique.

CONCLUSION

The main focus of the dissertation was to propose an architectural framework for incorporating data encryption along with access control to an IoT environment. Security and privacy of IoT is one of the most challenging and complicated aspects in the IoT domain due to the nature of these ubiquitous resource constrained devices. In our research we have shown that it is feasible for the IoT devices to have the capabilities to integrates security components with the middleware for securing the PII data of the users along with access control, encryption and integrity. In order to achieve the goal of the research we have used ABE as our cryptographic solution and also extended the architecture to reduce the overhead computation in the IoT middleware.

An overview of the security and privacy concerns of IoT are presented in Introduction where we highlighted the need for security of data, based on different aspects like the cloud, the data in transit, the trend of the IoT and including some vulnerabilities concerning the IoT technologies. We have shown the necessity of protecting the PII data based on the regulations and standards imposed by the governing bodies. The current trend of the IoT is not offering adequate security mechanism that deals with the data privacy, data encryption and access control.

In Chapter 1, we presented the details of the cryptographic primitives for the asymmetric cryptography that we have used in this dissertation including the PBC, ECC and ABE. In this chapter we have also presented a detailed background studies required for the IoT middlewares along with some of the well-known examples of IoT middlwares along with their comparison. In Chapter 2, we have illustrated the related works that have accomplished in the field of IoT security and privacy along with the ABE schemes and types and different variations of the schemes.

The proposed security architectural framework that is incorporated with the openHAB middleware was presented in Chapter 3. We have used a smart home environment to test and

implement the security architecture using ABE. Later we experimented our architecture with different ABE schemes to see the overhead of the implementations with respect to the resource consumptions like CPU and memory usage and as well the latency of the system based on the frequency of the data generated by the sensors in the smart home environment. From the experimental results, we figured that the overall overhead and latency is partially suitable for the IoT environment.

Then we extend the our privacy preserving architecture in Chapter 4 to outsource the encryption process to reduce the overhead in the smart home gateway. In this extended architecture, we proposed an ABE technique where the gateway performs partial encryption and the rest of the encryption computation was transferred to a proxy server. The main idea of this approach is to do a encryption with one attribute at the gateway and the rest of the computation is done at the proxy server. We have provided in details how the architecture works as well the experimentation and experimental setup for the evaluating the implementation with different ABE schemes. From the implementation and results, we have shown that we are capable of reducing 30% of the resource consumption on the smart gateway.

In Chapter 5, we have showed how it is possible to extend the our implementation for future directions using Machine Learning techniques and neural network model. In this chapter we have designed an architecture where every devices in the smart environment will act as encryption modules to reduce the overhead and ultimately the latency of the data from the source to the destination.

In our research we have shown that it is feasible to incorporate ABE in an smart home environment. Also we observed that CP-ABE is more dynamic in terms of access control but it requires more resources than other scheme. Whereas YCT has better performance since it does pre-computation during the initialization stage but it does not allow adding new attributes

dynamically. The decision for the ABE scheme for the smart home will be dependant on the parameters of the ABE like the attribute list and policy the length.

APPENDIX I

ATTRIBUTE BASED ENCRYPTION ALGORITHMS

1. CP-ABE

1.1 Setup

```
g, gp = group.random(G1), group.random(G2)
alpha, beta = group.random(ZR), group.random(ZR)
g.initPP(); gp.initPP()
h = g ** beta; f = g ** ~beta
e_gg_alpha = pair(g, gp ** alpha)
pk = { 'g':g, 'g2':gp, 'h':h, 'f':f,
'e_gg_alpha':e_gg_alpha }
mk = {'beta':beta, 'g2_alpha':gp ** alpha }
return(pk,mk)
```

1.2 Key Generation

```
r = group.random()
g_r = (pk['g2'] ** r)
D = (mk['g2_alpha'] * g_r) ** (1 / mk['beta'])
D_j, D_j_pr = {}, {}
for j in S:
    r_j = group.random()
    D_j[j] = g_r * (group.hash(j, G2) ** r_j)
    D_j_pr[j] = pk['g'] ** r_j
return { 'D':D, 'Dj':D_j, 'Djp':D_j_pr, 'S':S }
```

1.3 Encryption

```
policy = util.createPolicy(policy_str)
a_list = util.getAttributeList(policy)
s = group.random(ZR)
shares = util.calculateSharesDict(s, policy)
C = pk['h'] ** s
C_y, C_y_pr = {}, {}
for i in shares.keys():
    j = util.strip_index(i)
    C_y[i] = pk['g'] ** shares[i]
    C_y_pr[i] = group.hash(j, G2) ** shares[i]
return { 'C_tilde':(pk['e_gg_alpha'] ** s) * M,
        'C':C, 'Cy':C_y, 'Cyp':C_y_pr, 'policy':policy_str,
        'attributes':a_list }
```

1.4 Decryption

```
policy = util.createPolicy(ct['policy'])
pruned_list = util.prune(policy, sk['S'])
if pruned_list == False:
  return False
z = util.getCoefficients(policy)
A = 1
for i in pruned_list:
    j = i.getAttributeAndIndex(); k = i.getAttribute()
    A *= ( pair(ct['Cy'][j], sk['Dj'][k]) / pair(sk['Djp'][k],
    ct['Cyp'][j]) ) ** z[j]
return ct['C_tilde'] / (pair(ct['C'], sk['D']) / A)
```

2. KP-ABE

2.1 Setup

```
alpha1, alpha2, b = group.random(ZR), group.random(ZR),
                                group.random(ZR)
alpha = alpha1 * alpha2
g_G1, g_G2 = group.random(G1), group.random(G2) # PK 1,2
h_G1, h_G2 = group.random(G1), group.random(G2) # PK 3
g1b = g_G1 ** b
e_gg_alpha = pair(g_G1,g_G2) ** alpha
pk = { 'g_G1':g_G1, 'g_G2':g_G2, 'g_G1_b':g1b,
      'g_G1_b2':g1b ** b, 'h_G1_b':h_G1 ** b,
      'e(gg)_alpha':e_gg_alpha }
mk = { 'alpha1':alpha1, 'alpha2':alpha2, 'b':b,
         'h_G1':h_G1, 'h_G2':h_G2 }
return (pk, mk)
```

2.2 Key Generation

```
policy = util.createPolicy(policy_str)
attr_list = util.getAttributeList(policy)
s = mk['alpha1']; secret = s
shares = util.calculateSharesDict(secret, policy)
D = { 'policy': policy_str }
for x in attr_list:
    y = util.strip_index(x)
    d = []; r = group.random(ZR)
    if not self.negatedAttr(x): # meaning positive
        d.append((pk['g_G1'] ** (mk['alpha2'] * shares[x]))
```

```
        * (group.hash(y, G1) ** r))
        d.append((pk['g_G2'] ** r))
    D[x] = d
return D
```

2.3 Encryption

```
t = group.init(ZR, 0)
s = group.random(); sx = [s]
for i in range(len(attr_list)):
    sx.append(group.random(ZR))
    sx[0] -= sx[i]
    E3 = {}
    for i in range(len(attr_list)):
        attr = attr_list[i]
        E3[attr] = group.hash(attr, G1) ** s
    E1 = (pk['e(gg)_alpha'] ** s) * M
    E2 = pk['g_G2'] ** s
return {'E1':E1, 'E2':E2, 'E3':E3, 'attributes':attr_list }
```

2.4 Decryption

```
policy = util.createPolicy(D['policy'])
attrs = util.prune(policy, E['attributes'])
if attrs == False:
   return False
coeff = util.getCoefficients(policy)
Z = {}; prodT = 1
for i in range(len(attrs)):
    x = attrs[i].getAttribute()
```

```
    y = attrs[i].getAttributeAndIndex()
    if not self.negatedAttr(y):
        Z[y] = pair(D[y][0], E['E2'])
                / pair(E['E3'][x], D[y][1])
        prodT *= Z[y] ** coeff[y]
return E['E1'] / prodT
```

3. YCT-ABE

3.1 Setup

```
s = group.random(ZR)
g = group.random(G1)
self.attributeSecrets = {}
self.attribute = {}
for attr in attributes:
    si = group.random(ZR)
    self.attributeSecrets[attr] = si
    self.attribute[attr] = g**si
return (g**s, s) # (pk, mk)
```

3.2 Key Generation

```
policy = util.createPolicy(policy_str)
attr_list = util.getAttributeList(policy)
s = mk
shares = util.calculateSharesDict(s, policy)
D = { 'policy': policy_str, 'Du': d }
for x in attr_list:
    y = util.strip_index(x)
```

```
    d[y] = shares[x]/self.attributeSecrets[y]
return D
```

3.3 Encryption

```
k = group.random(ZR);
Cs = pk ** k
Ci = {}
for attr in attr_list:
    Ci[attr] = self.attribute[attr] ** k
symcrypt = SymmetricCryptoAbstraction(extractor(Cs))
C = symcrypt.encrypt(M)
return { 'C': C, 'Ci': Ci, 'attributes': attr_list }
```

3.4 Decryption

```
policy = util.createPolicy(D['policy'])
attrs = util.prune(policy, C['attributes'])
if attrs == False:
return False
coeff = util.getCoefficients(policy)
Z = {}
prodT = 1
for i in range(len(attrs)):
    x = attrs[i].getAttribute()
    y = attrs[i].getAttributeAndIndex()
    Z[y] = C['Ci'][x] ** D['Du'][x]
    prodT *= Z[y] ** coeff[y]
symcrypt = SymmetricCryptoAbstraction(extractor(prodT))
return symcrypt.decrypt(C['C'])
```

APPENDIX II

OPENHAB CONFIGURATION

1. Sensor Configuration Settings

```
String LTemp "Temperature: [%s]"
        {tcp="<[*:25002:'REGEX((.*))']"}
String LLight "Light: [%s]"
        {tcp="<[*:25003:'REGEX((.*))']"}
String KTemp "Temperature: [%s]"
        {tcp="<[*:25004:'REGEX((.*))']"}
String KLight "Light: [%s]"
        {tcp="<[*:25005:'REGEX((.*))']"}
String KSTemp "Stove Temperature: [%s]"
        {tcp="<[*:25006:'REGEX((.*))']"}
String KCont "Cabinet 1: [%s]"
        {tcp="<[*:25007:'REGEX((.*))']"}
String KGas "Smoke Detector: [%s]"
        {tcp="<[*:25008:'REGEX((.*))']"}
```

2. Rules for sensor classification

```
import org.openhab.core.library.types.*
var String Epath = "python3 /home/rc/Desktop/Research
/charm-dev/charm/schemes/abenc/Demo/encry.py "
var String Dpath = "python3 /home/rc/Desktop/Research
/charm-dev/charm/schemes/abenc/Demo/disp.py "
var String DEpath = "python3 /home/rc/Desktop/Research
/charm-dev/charm/schemes/abenc/Demo/decry.py "
var String path= "python3 /home/rc/Desktop/Research
```

```
/charm-dev/charm/schemes/abenc/Demo/update.py "

rule "data for sensor Ltemp"
when
                Item LTemp received update
        then
                var state = LTemp.state
                var String data
                var String recv
                var String decrypt
                var String encrypt = Epath +state+" LTemp"
                executeCommandLine(encrypt)
                recv = Dpath+"LTemp"
                data = executeCommandLine(recv,5000)
                ELTemp.postUpdate(data)
                decrypt = DEpath +"s1 LTemp"
                data = executeCommandLine(decrypt,5000)
                U1LTemp.postUpdate(data)
                decrypt = DEpath +"s2 LTemp"
                data = executeCommandLine(decrypt,5000)
                U2LTemp.postUpdate(data)
                decrypt = DEpath +"s3 LTemp"
                data = executeCommandLine(decrypt,5000)
                U3LTemp.postUpdate(data)
end
rule "data for sensor LLight"
when
                Item LLight received update
        then
```

```
                var state = LLight.state
                var String data
                var String recv
                var String decrypt
                var String encrypt = Epath +state+" LLight"
                executeCommandLine(encrypt)
                recv = Dpath+"LLight"
                data = executeCommandLine(recv,5000)
                ELLight.postUpdate(data)
                decrypt = DEpath +"s1 LLight"
                data = executeCommandLine(decrypt,5000)
                U1LLight.postUpdate(data)
                decrypt = DEpath +"s2 LLight"
                data = executeCommandLine(decrypt,5000)
                U2LLight.postUpdate(data)
                decrypt = DEpath +"s3 LLight"
                data = executeCommandLine(decrypt,5000)
                U3LLight.postUpdate(data)
end
rule "data for sensor KTemp"
when
                Item KTemp received update
        then
                var state = KTemp.state
                var String data
                var String recv
                var String decrypt
                var String encrypt = Epath +state+" KTemp"
                executeCommandLine(encrypt)
```

```
                recv = Dpath+"KTemp"
                data = executeCommandLine(recv,5000)
                EKTemp.postUpdate(data)
                decrypt = DEpath +"s1 KTemp"
                data = executeCommandLine(decrypt,5000)
                U1KTemp.postUpdate(data)
                decrypt = DEpath +"s2 KTemp"
                data = executeCommandLine(decrypt,5000)
                U2KTemp.postUpdate(data)
                decrypt = DEpath +"s3 KTemp"
                data = executeCommandLine(decrypt,5000)
                U3KTemp.postUpdate(data)
end
rule "data for sensor KLight"
when
                Item KLight received update
        then
                var state = KLight.state
                var String data
                var String recv
                var String decrypt
                var String encrypt = Epath +state+" KLight"
                executeCommandLine(encrypt)
                recv = Dpath+"KLight"
                data = executeCommandLine(recv,5000)
                EKLight.postUpdate(data)
                decrypt = DEpath +"s1 KLight"
                data = executeCommandLine(decrypt,5000)
                U1KLight.postUpdate(data)
```

```
                decrypt = DEpath +"s2 KLight"
                data = executeCommandLine(decrypt,5000)
                U2KLight.postUpdate(data)
                decrypt = DEpath +"s3 KLight"
                data = executeCommandLine(decrypt,5000)
                U3KLight.postUpdate(data)
end
```

3. Admin Control Panel

```
sitemap admin label= "CONTROL PANEL"{
        Frame label="Living Room Temperature"{
                Switch item= LTempa1
                Switch item= LTempa2
                Switch item= LTempa3
                Switch item= LTempa4
                Switch item= LTempa5
                Switch item= LTempa6

        }
        Frame label="Living Room Light"{
                Switch item= LLighta1
                Switch item= LLighta2
                Switch item= LLighta3
                Switch item= LLighta4
                Switch item= LLighta5
                Switch item= LLighta6
        }
        Frame label="Kitchen Temperature"{
                Switch item= KTempa1
```

```
                Switch item= KTempa2
                Switch item= KTempa3
                Switch item= KTempa4
                Switch item= KTempa5
                Switch item= KTempa6
        }
        Frame label="Kitchen Light"{
                Switch item= KLighta1
                Switch item= KLighta2
                Switch item= KLighta3
                Switch item= KLighta4
                Switch item= KLighta5
                Switch item= KLighta6
        }

        }
        Frame label="Policy"{
                Webview url= "http://localhost:8080
                /static/form/form.html" height= 10
        }
}
```

4. Example of user diaplay

```
sitemap default label="Service 1" {
        Frame label= "Living Room"
        {
        Text item= U1LTemp icon="temperature"
        Text item= U1LLight icon="light-on"
        }
```

```
        Frame label= "Kitchen"{
        Text item= U1KTemp icon="temperature"
        Text item= U1KLight icon="light-on"
        Text item= U1KSTemp icon="temperature"
        Text item= U1KCont icon="contact"
        Text item= U1KGas icon="smoke"
        }
        Frame label="BedRoom 1"{
        Text item= U1R1Temp icon="temperature"
        Text item= U1R1Light icon="light-on"
        Text item= U1R1Cont icon="contact"
        }
        Frame label="BedRoom 2"{
        Text item= U1R2Temp icon="temperature"
        Text item= U1R2Light icon="light-on"
        Text item= U1R2Cont icon="contact"
        }
        Frame label="Bathroom"{
        Text item= U1BTemp icon="temperature"
        Text item= U1BLight icon="light-on"
        Text item= U1BCont icon="contact"
        }
        Frame label="Utilities"{
        Text item= U1UElec icon="energy"
        Text item= U1UWater icon="water1"
        Text item= U1UGas icon="gas"
        }
}
```

BIBLIOGRAPHY

Akinyele, J. A., Garman, C., Miers, I., Pagano, M. W., Rushanan, M., Green, M. & Rubin, A. D. (2013). Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2), 111–128.

Almorsy, M., Grundy, J. & Müller, I. (2016). An analysis of the cloud computing security problem. *arXiv preprint arXiv:1609.01107*.

Ambrosin, M., Conti, M. & Dargahi, T. (2015). On the feasibility of attribute-based encryption on smartphone devices. *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*, pp. 49–54.

Ambrosin, M., Anzanpour, A., Conti, M., Dargahi, T., Moosavi, S. R., Rahmani, A. M. & Liljeberg, P. (2016). On the feasibility of attribute-based encryption on internet of things devices. *IEEE Micro*, 36(6), 25–35.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M. et al. (2017). Understanding the mirai botnet. *USENIX Security Symposium*.

Ashton, K. (2016). Q and A With The Father of IoT Kevin Ashton. Consulted at https://iotuk.org.uk/qa-with-the-father-of-iot-kevin-ashton/.

Assistant, H. (2018). Home Assistant is an open-source home automation platform. Consulted at https://www.home-assistant.io/.

AT&T. (2016). IoT evolution: Security trails deployment. Consulted at https://www.business.att.com/cybersecurity/archives/v2/iot/.

Avoyan, H. (2017). Three Types of Cloud Computing Services. Consulted at http://www.monitis.com/blog/3-types-of-cloud-computing-services/.

Balamurugan, B., Nirmala Devi, M., Meenakshi, R. & Abinaya, V. (2013). Cipher-text Outsourced Decryption with Enhanced Access Rights. *International Conference on Mathematical Computer Engineering-ICMCE*, pp. 1226.

Barnes, C. J. (2017). Smart Home Alone: The Worlds Gateway to More Efficient Use of Energy and Mayhem. Consulted at https://digitalcommons.law.lsu.edu/jelr/vol5/iss2/10.

BBC. (2017). Smart home devices used as weapons in website attack. Consulted at http://www.techrepublic.com/article/here-are-the-biggest-iot-security-threats-facing-the-enterprise-in-2017/.

Bethencourt, J., Sahai, A. & Waters, B. (2007). Ciphertext-policy attribute-based encryption. *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pp. 321–334.

Borgh, J., Ngai, E., Ohlman, B. & Malik, A. M. (2017, June). Employing attribute-based encryption in systems with resource constrained devices in an information-centric networking context. *2017 Global Internet of Things Summit (GIoTS)*, pp. 1-6. doi: 10.1109/GIOTS.2017.8016277.

Brodkin, J. (2008). Gartner: Seven cloud-computing security risks. Consulted at https://www.infoworld.com/article/2652198/security/gartner--seven-cloud-computing-security-risks.html.

Cao, H. & Cai, J. (2018). Distributed Multiuser Computation Offloading for Cloudlet-Based Mobile Cloud Computing: A Game-Theoretic Machine Learning Approach. *IEEE Transactions on Vehicular Technology*, 67(1), 752–764.

Chai, Q. & Gong, G. (2012, June). Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. *2012 IEEE International Conference on Communications (ICC)*, pp. 917-922. doi: 10.1109/ICC.2012.6364125.

Charm. [Accessed: 01-29-2017]. Charm: A tool for rapid cryptographic prototyping. Consulted at https://github.com/JHUISI/charm.

Chen, X., Jiao, L., Li, W. & Fu, X. (2016). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24(5), 2795–2808.

Cisco. (2017). Complete Visual Networking Index (VNI) Forecast. Consulted at https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html#~complete-forecast.

Erika McCallister, T. G. & Scarfone, K. (2010). NIST Guide to Protecting the Confidentiality of Personally Identifiable Information (PII). Consulted at https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-122.pdf.

Fernandes, E., Rahmati, A., Eykholt, K. & Prakash, A. (2017). Internet of Things Security Research: A Rehash of Old Ideas or New Intellectual Challenges? *IEEE Security Privacy*, 15(4), 79-84. doi: 10.1109/MSP.2017.3151346.

Ferraiolo, D., Cugini, J. & Kuhn, D. R. (1995). Role-based access control (RBAC): Features and motivations. *Proceedings of 11th annual computer security application conference*, pp. 241–48.

FISMA. (2016). Federal Information Security Management Act. Consulted at https://www.dhs.gov/fisma.

FTC. (2016). FTC Privacy and Data Security. Consulted at https://www.ftc.gov/reports/privacy-data-security-update-2016.

Gartner. (2017a). Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. Consulted at https://www.gartner.com/newsroom/id/3598917.

Gartner. (2017b). Gartner Information Security Spending 2017. Consulted at https://www.gartner.com/newsroom/id/3784965.

GDPR. (2016). EU General Data Protection Regulation. Consulted at https://www.eugdpr.org/.

Goyal, V., Pandey, O., Sahai, A. & Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89–98.

Green, M., Hohenberger, S. & Waters, B. (2011). Outsourcing the Decryption of ABE Ciphertexts. *USENIX Security Symposium*, 2011(3).

Henze, M., Hummen, R., Matzutt, R. & Wehrle, K. (2014). A trust point-based security architecture for sensor data in the cloud. In *Trusted Cloud Computing* (pp. 77–106). Springer.

Henze, M., Hermerschmidt, L., Kerpen, D., Häußling, R., Rumpe, B. & Wehrle, K. (2016). A comprehensive approach to privacy in the cloud-based Internet of Things. *Future Generation Computer Systems*, 56, 701–718.

HomeGenie. (2018). The open source, programmable, home automation server for smart connected devices and applications. Consulted at http://www.homegenie.it/.

Institute, P. (2018). Data Breach Report in USA 2017. Consulted at https://www.ponemon.org/blog/2017-cost-of-data-breach-study-united-states.

Ishiguro, T., Kiyomoto, S. & Miyake, Y. (2013). A key-revocable attribute-based encryption for mobile cloud environments. *Security and Cryptography (SECRYPT), 2013 International Conference on*, pp. 1–11.

Jin, Y., Tian, C., He, H. & Wang, F. (2015, Aug). A Secure and Lightweight Data Access Control Scheme for Mobile Cloud Computing. *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*, pp. 172-179. doi: 10.1109/BDCloud.2015.57.

Jung, M., Kienesberger, G., Granzer, W., Unger, M. & Kastner, W. (2011). Privacy enabled web service access control using SAML and XACML for home automation gateways. *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, pp. 584–591.

Kattepur, A., Dohare, H., Mushunuri, V., Rath, H. K. & Simha, A. (2016). Resource Constrained Offloading in Fog Computing. *Proceedings of the 1st Workshop on Middleware for Edge Clouds & Cloudlets*, pp. 1.

Kemp, R., Palmer, N., Kielmann, T. & Bal, H. (2010). Cuckoo: a computation offloading framework for smartphones. *International Conference on Mobile Computing, Applications, and Services*, pp. 59–79.

Khan, M. A. (2015). A survey of computation offloading strategies for performance improvement of applications running on mobile devices. *Journal of Network and Computer Applications*, 56, 28–40.

Kovachev, D., Yu, T. & Klamma, R. (2012). Adaptive computation offloading from mobile devices into the cloud. *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pp. 784–791.

Lai, J., Deng, R. H., Guan, C. & Weng, J. (2013). Attribute-based encryption with verifiable outsourced decryption. *IEEE Transactions on information forensics and security*, 8(8), 1343–1354.

Maddox, T. (2016). Here are the biggest IoT security threats facing the enterprise in 2017. Consulted at https://www.techrepublic.com/article/here-are-the-biggest-iot-security-threats-facing-the-enterprise-in-2017/.

Maletsky, K. (2015). RSA vs ECC comparison for embedded systems. *White Paper, Atmel*, 5.

Malina, L., Hajny, J., Fujdiak, R. & Hosek, J. (2016). On perspective of security and privacy-preserving solutions in the internet of things. *Computer Networks*, 102, 83–95.

Marin, A., Mueller, W., Schaefer, R., Almenárez, F., Díaz, D. & Ziegler, M. (2007). Middleware for secure home access and control. *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, pp. 489–494.

Mazza, D., Tarchi, D. & Corazza, G. E. (2016). A cluster based computation offloading technique for mobile cloud computing in smart cities. *Communications (ICC), 2016 IEEE International Conference on*, pp. 1–6.

Mbed, A. (2018). The Arm Mbed IoT Device Platform. Consulted at https://www.mbed.com/en/.

McKendrick, J. (2016). With Internet Of Things And Big Data, 92In The Cloud. Consulted at https://www.forbes.com/sites/joemckendrick/2016/11/13/with-internet-of-things-and-big-data-92-of-everything-we-do-will-be-in-the-cloud/#1e9c6abb4ed5.

Miyaji, A., Nakabayashi, M. & Takano, S. (2000). Characterization of elliptic curve traces under FR-reduction. *International Conference on Information Security and Cryptology*, pp. 90–108.

Moncrieff, S., Venkatesh, S. & West, G. (2007). Dynamic privacy in a smart house environment. *Multimedia and Expo, 2007 IEEE International Conference on*, pp. 2034–2037.

Morgan, S. (2018). Cybercrime Report 2017. Consulted at https://cybersecurityventures.com/2015-wp/wp-content/uploads/2017/10/2017-Cybercrime-Report.pdf.

Newman, D. (2017). The Top 8 IoT Trends For 2018. Consulted at https://www.forbes.com/sites/danielnewman/2017/12/19/the-top-8-iot-trends-for-2018/#649e5c1067f7.

openHAB. (2018). A vendor and technology agnostic open source automation software for your home. Consulted at https://www.openhab.org/.

OpenRemote. (2018). OpenRemote is the Open Source Middleware for the Internet of Things. Consulted at http://www.openremote.com/.

Osborne, C. (2017). FDA issues recall of 465,000 St. Jude pacemakers to patch security holes. Consulted at https://www.zdnet.com/article/fda-forces-st-jude-pacemaker-recall-to-patch-security-vulnerabilities/.

Oualha, N. & Nguyen, K. T. (2016). Lightweight attribute-based encryption for the Internet of Things. *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*, pp. 1–6.

Peter Middleton,Tracy Tsai, M. Y. A. G. & Rueb, D. (2017). Forecast: Internet of Things — Endpoints and Associated Services, Worldwide, 2017. Consulted at https://www.gartner.com/doc/3840665/forecast-internet-things--endpoints.

Prinzlau, M. (2017). Six security risks of enterprises using cloud storage and file sharing application. Consulted at https://digitalguardian.com/blog/6-security-risks-enterprises-using-cloud-storage-and-file-sharing-apps.

Punia, A., Gupta, D. & Jaiswal, S. (2017). A perspective on available security techniques in IoT. *Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017 2nd IEEE International Conference on*, pp. 1553–1559.

Qin, B., Deng, R. H., Liu, S. & Ma, S. (2015). Attribute-based encryption with efficient verifiable outsourced decryption. *IEEE Transactions on Information Forensics and Security*, 10(7), 1384–1393.

Razzaque, M. A., Milojevic-Jevric, M., Palade, A. & Clarke, S. (2016). Middleware for internet of things: a survey. *IEEE Internet of Things Journal*, 3(1), 70–95.

Ródenas, S. (2015). El Internet of Things desde 5 startups. Consulted at http://www.sociedaddelainnovacion.es/revolucion-internet-of-things-desde-5-startups/.

Roberto Minerva, A. B. & Rotondi, D. (2015). Towards a definition of the Internet of Things (IoT). Consulted at https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf.

Sahai, A. & Waters, B. (2005). Fuzzy identity-based encryption. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473.

Samie, F., Tsoutsouras, V., Bauer, L., Xydis, S., Soudris, D. & Henkel, J. (2016). Computation offloading and resource allocation for low-power IoT edge devices. *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, pp. 7–12.

Schurgot, M. R., Shinberg, D. A. & Greenwald, L. G. (2015). Experiments with security and privacy in IoT networks. *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pp. 1–6.

Sharma, R. et al. (2017). Computation Offloading in Mobile Cloud Computing. *International Journal of Current Trends in Science and Technology*, 7(12), 20501–20510.

Shukla, R. M. & Munir, A. (2016). A computation offloading scheme leveraging parameter tuning for real-time IoT devices. *Nanoelectronic and Information Systems (iNIS), 2016 IEEE International Symposium on*, pp. 208–209.

Singh, J., Pasquier, T., Bacon, J., Ko, H. & Eyers, D. (2016). Twenty security considerations for cloud-supported Internet of Things. *IEEE Internet of Things Journal*, 3(3), 269–284.

Skerrett, I. (2017). IoT Developer Trends 2017 Edition. Consulted at https://ianskerrett.wordpress.com/2017/04/19/iot-developer-trends-2017-edition/.

Stanislav, M. & Beardsley, T. (2015). Hacking iot: A case study on baby monitor exposures and vulnerabilities. *Rapid7 Research, Tech. Report*.

Stankovic, J. A. (2014). Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1), 3–9.

Touati, L., Challal, Y. & Bouabdallah, A. (2014). C-cp-abe: Cooperative ciphertext policy attribute-based encryption for the internet of things. *Advanced Networking Distributed Systems and Applications (INDS), 2014 International Conference on*, pp. 64–69.

Tout, H., Talhi, C., Kara, N. & Mourad, A. (2017). Smart mobile computation offloading: Centralized selective and multi-objective approach. *Expert Systems with Applications*, 80, 1–13.

Tripathi, V. (2017). Adaptive Computation Offloading in Mobile Cloud Computing.

Wang, C., Liang, C., Yu, F. R., Chen, Q. & Tang, L. (2017). Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. *IEEE Transactions on Wireless Communications*, 16(8), 4924–4938.

Wang, X., Zhang, J., Schooler, E. M. & Ion, M. (2014). Performance evaluation of attribute-based encryption: Toward data privacy in the IoT. *Communications (ICC), 2014 IEEE International Conference on*, pp. 725–730.

Wikipedia. (2018). Personally identifiable information. Consulted at https://en.wikipedia.org/wiki/Personally_identifiable_information.

Wilbanks, L. (2007). The impact of personally identifiable information. *IT Professional*, 9(4).

WolframMathWorld. (2018). Elliptic Curve. Consulted at http://mathworld.wolfram.com/EllipticCurve.html.

Yao, X., Chen, Z. & Tian, Y. (2015). A lightweight attribute-based encryption scheme for the Internet of Things. *Future Generation Computer Systems*, 49, 104–112.

Yuan, E. & Tong, J. (2005). Attributed based access control (ABAC) for web services. *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*.

Zetter, K. (2012). Flaw in Home Security Cameras Exposes Live Feeds to Hackers. Consulted at https://www.wired.com/2012/02/home-cameras-exposed/.

Zhou, Z. & Huang, D. (2012). Efficient and secure data storage operations for mobile cloud computing. *Proceedings of the 8th International Conference on Network and Service Management*, pp. 37–45.