12-2018

Protecting Privacy of Data in the Internet of Things with Policy Enforcement Fog Module

Abduljaleel Al-Hasnawi
Western Michigan University, jaleelaldaly@yahoo.com

PROTECTING PRIVACY OF DATA IN THE INTERNET OF THINGS WITH POLICY ENFORCEMENT FOG MODULE

by

Abduljaleel Al-Hasnawi

A dissertation submitted to the Graduate College
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
Computer Science
Western Michigan University
December 2018

Doctoral Committee:

Ajay Gupta, Ph.D., Chair
Steve Carr, Ph.D.
Ala Al-Fuqaha, Ph.D.
Ikhlas Abdel-Qader, Ph.D.

ACKNOWLEDGMENTS

First and foremost, I would like to thank God Almighty for giving me the strength, knowledge, ability and opportunity to undertake this research study and to persevere and complete it satisfactorily. Without his blessings, this achievement would not have been possible.

I would like to express my deepest and sincere gratitude to my supervisors Dr. Ajay Gupta and Dr. Steve Carr. They have advised, inspired, mentored and supported me to achieve success in this dissertation work. Without their endurance, support, and assistance, the completion of this dissertation would not have been possible.

I take pride in acknowledging the insightful guidance of my committee members Dr. Ala Al-Fuqaha and Dr. Ikhlas Abdel-Qader for sparing their valuable time whenever I approached them and showing me the way ahead.

In my journey towards this degree, I have found a pillar of support in my Guide, Dr. Steve Carr, Chair of the Department of Computer Science. He has been there providing his heartfelt support and guidance at all times. I shall eternally be grateful to him for his assistance.

I would also like to express my gratitude to Dr. Susan Stapleton, Dean of the Graduate College, and Dr. Omran, the Iraqi Cultural Attaché, and the entire staff at Western Michigan University, as well as Computer Science office, who have been so helpful and cooperative in giving their support at all times to help me achieve my goal.

I would thank Dr. Leszek Lilien for his efforts and times working with me in the proposal stage and the publication work.

I have great pleasure in acknowledging my gratitude to my colleagues at Computer Science Department, their support and encouragement have been great contributors in the completion of the work. Special thanks going to my dear friend Ahmed Al-Gburi, who have, in his own way, kept me going on my path to success, assisting me as per his abilities, in whatever manner possible and for ensuring that good times keep flowing.

My acknowledgement would be incomplete without thanking the biggest source of my strength, my family. The blessings of my father and my mother, the love and care of my wife, and the support of my brothers and sisters, especially from my dear brother Wael, have all made a tremendous contribution in helping me reach this stage in my life. I thank them for putting up with me in difficult moments where I felt stumped and for goading me on to follow my dream of getting this degree. This would not have been possible without their unwavering and unselfish love and support given to me at all times.

I would like to dedicate this work to my father and my mother whose dreams for me have resulted in this achievement and without their loving upbringing and nurturing; I would not have been where I am today and what I am today.

I also would like to dedicate this work to the soul of my son Mohammed, I wish that you are with me in this time, you are always in my heart, I missed you so much.

Finally, my love to go to the source of my happiness my daughter Nada, my son Hussein, and to my cutest daughters Jenna & Jumana, I love you so much. For all of you, I hope a happy life and a best of luck in your future.

Abduljaleel Al-Hasnawi

PROTECTING PRIVACY OF DATA IN THE INTERNET OF THINGS WITH POLICY ENFORCEMENT FOG MODULE

Abduljaleel Al-Hasnawi, Ph.D.

Western Michigan University, 2018

The growth of IoT applications has resulted in generating massive volumes of data about people and their surroundings. Significant portions of these data are sensitive since they reflect peoples' behaviors, interests, lifestyles, etc. Protecting sensitive IoT data from privacy violations is a challenge since these data need to be handled by public networks, servers and clouds, most of which are untrusted parties for data owners. In this study, a solution called Policy Enforcement Fog Module (PEFM) is proposed for protecting sensitive IoT data. The primary task of the PEFM solution is mandatory enforcement of privacy policies for sensitive IoT data—whenever these data are accessed, throughout their entire lifecycle. The key feature of PEFM is its placement within the fog computing infrastructure, which assures that PEFM operates as closely as possible to data sources within the edge of the IoT network. PEFM enforces privacy policies directly for data accessed by local IoT applications, using components inherited from the eXtensible Access Control Markup Language (XACML) architecture. PEFM also assures enforcement of privacy policies for data accessed by remote IoT applications, using XACML and Active Data Bundles (ADBs) that can run on any visited host and enforce policies automatically for data accessed by these hosts.

The Foscam Home Surveillance System (FHSS) was selected as a proof-of-concept case study to test the capabilities of PEFM in protecting sensitive surveillance data. The privacy threats in FHSS are investigated, and the framework of using PEFM for FHSS to address these threats is proposed. Different scenarios are discussed regarding the privacy risk of having malicious insiders or attackers in the system for both local and remote data usages. A scenario with no risk is considered as a baseline with which a scenario with a certain level of privacy risk is compared.

To evaluate the performance of the proposed framework, a comprehensive simulation design, based on realistic FHSS configurations, is developed. Simulation experiments were implemented using SimPy, a process-based discrete-event simulation framework based on standard Python, running in the PyCharm, IDE environment. The experimental results are discussed in terms of the privacy goals achieved by PEFM and the corresponding system performance overhead introduced in terms of latency and throughput. Our results show that PEFM increases users' control for their data with the number of enforced privacy policies. However, the overhead introduced by enforcing increased polices should not exceed the threshold determined by the real-time constraints. We show that PEFM assures selective data disclosure with better performance than for the baseline mainly due to data minimization. Finally, the results indicate that better privacy controls with minimal overhead can be achieved if most PEFM processes are executed by the local fog nodes. Migrating parts of PEFM processes to remote fog nodes or the cloud incurs more overhead than using strictly local fog nodes. This overhead is the price to be paid for a higher level of privacy in terms of lifecycle data protection. So, there is a tradeoff between overhead and the desired level of privacy. The overhead should be acceptable by applications that are not time-sensitive with hard deadlines.

TABLE OF CONTENTS

ACKNOWLEDGMENTS ii

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER

1. INTRODUCTION 1

1.1. Background Information 1

1.1.1. IoT Reference Model 1
1.1.2. Fog Computing 4
1.1.3. IoT Data Privacy 8
1.1.4. Sensitive Data 8
1.1.5. Privacy Policies 9
1.1.6. Data Ownership 10
1.1.7. Selected IoT Privacy Threats 10
1.1.8. Privacy Controls in IoT 11
1.1.9. Active Data Bundle 12

1.2. Motivation 15

1.2.1. Motivation for Protecting Sensitive IoT Data 15
1.2.2. Motivation for Policy Enforcement 16
1.2.3. Motivation for Fog Computing 18

1.3. Problem Statement 19

1.4. Research Hypothesis 19

1.5. Research Scope 20

1.6. Main Contributions 21

1.7. Dissertation Organization 22

2. LITERATURE REVIEW 23

Table of Contents—Continued

CHAPTER

2.1. Encryption-Based Privacy Solutions 23

2.1.1. Public-Key Encryption.....23
2.1.2. Attribute-Based Encryption 24
2.1.3. Homomorphic Encryption 25
2.1.4. Point-to-Point Encryption 25

2.2. Identity Management-Based Privacy Solutions..... 26

2.2.1. K-Anonymity 26
2.2.2. Pseudonymity.....27
2.2.3. Attribute-Based Signature.....27
2.2.4. Group Signatures 28
2.2.5. Ring Signatures.....28

2.3. Policy Enforcement-Based Privacy Solutions 29

2.3.1. Mandatory Enforcement of Privacy Policies 29
2.3.2. Policy-Based Management 30
2.3.3. Cloudlet-Based Privacy Mediator.....30

2.4. Self-Protecting Based Privacy Solutions 31

2.4.1. DigiBox.....31
2.4.2. Agent-Based Informed Consent.....32
2.4.3. Personal Digital Rights Management 33

2.5. Comparison of Related Work with PEFM..... 33
2.6. Protecting Sensitive Data with Active Data Bundles 33

3. THE PROPOSED SOLUTION 38

3.1. Introduction..... 38
3.2. Overview of PEFM..... 38

3.2.1. Real-Time and Non-Real-Time Processing of IoT Data 39
3.2.2. Local and Remote Policy Enforcement 39

3.3. System Design 40

Table of Contents—Continued

CHAPTER

3.3.1. LPEM System Design 44

3.3.2. RPEM System Design 47

3.4. Chapter Conclusions 50

4. PROOF OF CONCEPT CASE STUDY 51

4.1. Introduction 51

4.2. Description of Foscam Home Surveillance System 51

4.2.1. Foscam IP Camera 52
4.2.2. Foscam Base Station 52

4.3. Privacy Threats in FHSS 53

4.4. Framework of Using PEFM for FHSS 53

4.5. Scenarios of the FHSS-based PEFM Framework 55

4.5.1. Local Foscam Scenarios 55
4.5.2. Remote Foscam Scenarios 58

4.6. Chapter Conclusion 61

5. SIMULATION DESIGN 62

5.1. Introduction 62
5.2. Simulation Assumptions 62
5.3. Simulation Parameters 64

5.3.1. Parameters for Network Nodes 64
5.3.2. Parameters for Network Links 65
5.3.3. Parameters for Network Configurations 66

5.4. Random Simulation Variables 67

5.4.1. Random Variables for Packets 67
5.4.2. Random Variables for Selective Data Disclosure 69
5.4.3. Random Variables for Privacy Policy 70

Table of Contents—Continued

CHAPTER

5.5. System Performance Measures 70

5.5.1. Latency 71
5.5.2. Throughput 73

5.6. Privacy Control Measures 74

5.6.1. Privacy Risk Control 74
5.6.2. User Control Level 75
5.6.3. Selective Data Disclosure 75

5.7. Chapter Conclusion 76

6. PERFORMANCE EVALUATION 77

6.1. Introduction 77
6.2. Simulation Implementation 78
6.3. Experimental Setup 80
6.4. Experimental Results for Privacy Risk Control 81

6.4.1. LPEM Privacy Risk Control and System Performance 81
6.4.2. RPEM Privacy Risk Control and System Performance 86

6.5. Experimental Results for Selective Data Disclosure 91

6.5.1. LPEM Selective Data Disclosure and System Performance 91
6.5.2. RPEM Selective Data Disclosure and System Performance 94

6.6. Experimental Results for User Control 98

6.6.1. LPEM User Control and System Performance 98
6.6.2. RPEM User Control and System Performance 100

6.7. Chapter Conclusion 101

7. CONCLUSIONS AND FUTURE WORK 102

7.1. Conclusions 102

7.2. Future Work 105

8. REFERENCES 106

LIST OF TABLES

2.1. A comparison of the related work with PEFM in terms of privacy goals 34

5.1. Simulation parameters for network nodes 64

5.2. Simulation parameters for network links 65

5.3. Parameters for network configurations 67

5.4. Random variables for Packets. 69

5.5. Random variables for selective data disclosure 70

LIST OF FIGURES

1.1. IoT Reference Model [1, 2, 3]. 4

1.2. Fog Computing Architecture [7]. 5

1.3. Fog Node Structure [8]. 7

1.4. Active Data Bundle [17, 24]. 13

3.1. The high-level structure of PEFM within EFN and its IoT system. 41

3.2. A high-level flow diagram for PEFM process. 43

3.3. LPEM Structure and process flow. 45

3.4. Flow diagram for LPEM process. 46

3.5. RPEM structure and process flow. 48

3.6. Flow diagram for RPEM process. 49

4.1. Framework of using PEFM for FHSS. 54

4.2. Home no-risk scenario. 56

4.3. Home malicious insider scenario. 57

4.4. Home attacker scenario. 57

4.5. Home high-risk scenario. 58

4.6. Cloud no-risk scenario. 59

4.7. Cloud malicious insider scenario. 59

4.8. Cloud attacker scenario. 60

4.9. Cloud high-risk scenario. 60

6.1. Absolute and relative LPEM latency for privacy risk control. 82

List of Figures—Continued

6.2. LPEM Latency gain for privacy risk control. 82

6.3. Absolute and relative LPEM throughput for privacy risk control. 85

6.4. LPEM throughput gain and overhead for privacy risk control. 85

6.5. Absolute and relative RPEM latency for privacy risk control. 88

6.6. RPEM Latency overhead for privacy risk control. 88

6.7. Absolute and relative RPEM throughput for privacy risk control. 90

6.8. RPEM throughput overhead for privacy risk control 90

6.9. LPEM selective data disclosure for the lowest sensitivity level. 92

6.10. LPEM selective data disclosure for moderate sensitivity levels. 93

6.11. LPEM selective data disclosures for the highest sensitivity level. 94

6.12. LPEM latency gain by selective data disclosure. 94

6.13. RPEM selective data disclosure for the lowest sensitivity level. 95

6.14. RPEM selective data disclosure for moderate sensitivity levels. 96

6.15. RPEM selective data disclosures for the highest sensitivity level. 97

6.16. RPEM latency overhead by selective data disclosure. 97

6.17. LPEM's hard real-time and soft real-time deadlines for response time. 99

6.18. LPEM response times for increasing user control. 99

6.19. RPEM's hard real-time and soft real-time deadlines for response time. 100

6.20. RPEM response times for increasing user control. 100

CHAPTER 1

INTRODUCTION

This chapter starts by providing background information on the Internet of Things (IoT) reference model, Fog computing, data privacy in IoT, and an overview of using Active Data Bundles (ADBs) for data privacy. The chapter then presents the motivation for protecting sensitive IoT data as well as the motivation for using a policy enforcement approach to achieve that protection. Next, it states the problem statement and the research hypothesis, as well as the scope and the limitations of the proposed research. Finally, the chapter outlines the main dissertation contributions and the organization of the dissertation.

1.1. Background Information

1.1.1. IoT Reference Model

To facilitate IoT analysis and design, various IoT reference models were developed. In this research, we adopt the four-layer IoT reference model, proposed by us after studying other IoT reference models [1], [2], [3] and shown in Figure 1.1. It consists of four main layers with sublayers as discussed below in the order from the bottom up:

1) *Perception Layer*: IoT requires a comprehensive perception of the environment, this layer obtains information about objects anytime and anywhere. It consists of two sublayers: (a) *device sublayer* includes a large collection of heterogeneous physical sensing devices, actuators, and controllers. These are typically thin clients' devices, with low processing

and storage capabilities. They can generate data, collect data, respond to queries, communicate among themselves and with their environment, and be controlled over the network. Typically, data collected in this layer pertain to location, orientation, object motion, temperature, humidity, air pollution, etc.; and (b) *connectivity sublayer* includes all the communication technologies required to enable connectivity between IoT devices at the perception layer. The primary function of this sublayer is the real-time transmission of sensed data to the processing points to enable real-time decision making.

2) *Network Layer* includes all hardware and software entities for communication networks. The primary function of this layer is to assure reliable and timely data transmission. This layer includes converting network data flows into information suitable for higher levels of processing. This layer consists of two sublayers: (a) *access gateway sublayer* includes a network infrastructure that enables communication among IoT devices as well as communication among devices and the core network sublayer, and (b) *core network sublayer* manages all IoT communications. It manages communication among different IoT service entities, among gateways and the network, across the network, as well as among the network and data processing entities. It includes the implementation of various communication protocols like MQTT, XMPP, AMQP, and DDS [4].

3) *Support Layer* includes middleware technologies that implement IoT services and perform data abstraction; both are needed to make data usable for IoT applications. This layer consists of two sublayers: (a) *service sublayer* includes all middleware technologies that implement IoT services, including computation, analysis, storage, virtualization, etc. [5]. It supports many types of data processing—such as data aggregation, data accumulation, and high-volume data analysis. The primary functions of this layer are reformatting

network packets into relational database tables and transforming event-based computing into query-based computing; both are needed to make data easily accessible to the IoT application layer; and (b) *data abstraction sublayer* includes consolidating data from multiple sources, simplifying them to create schemas and views suitable for applications. The primary functions of this level are reconciling different formats, patterns, semantics, etc., of data coming from various resources; confirming data consistency for the higher-level applications; as well as normalizing and indexing data to assure fast application access.

4) *Application Layer* provides global management for applications, based on information processed in the support layer. It is considered the hub of services requested by end users. The primary function of this layer is providing smart services to the end IoT users. This layer consists of two sublayers: (a) *interface sublayer* enables IoT users to use and customize different IoT applications by enabling user-application interactions and the management of the IoT things. An interface profile (IFP) is an example of a service provided by this sublayer; IFP is a subset of service standards that support interaction with applications deployed on the IoT network [6]; and (b) *core application sublayer* allows people to handle their data, collaborate with other people, meet their needs, and improve their life. The primary functions of this sublayer are building business logic models to empower end users to do their work, and providing the services requested by IoT users. This sublayer supports diverse collaborative processes enabled by core IoT applications. Examples of such processes are decision-making, evaluating, monitoring, and developing processes.

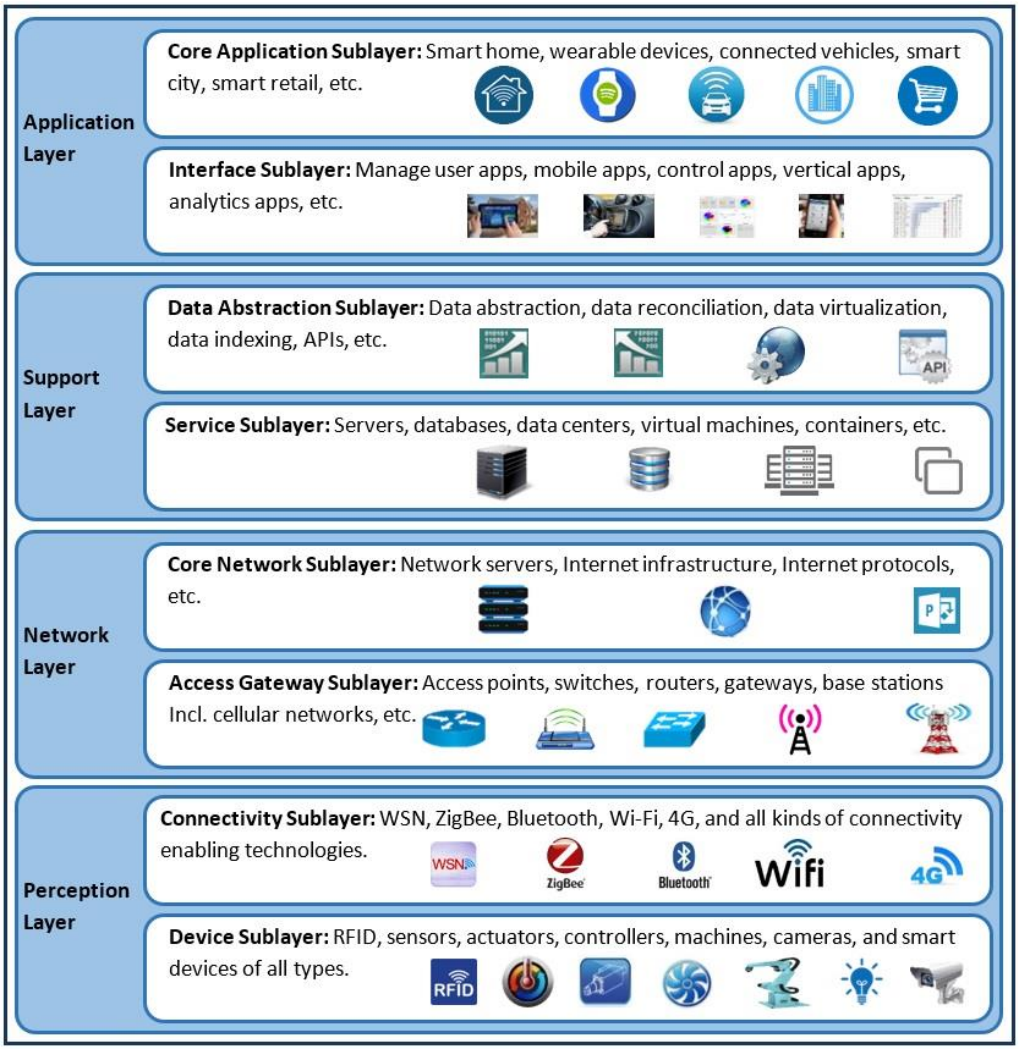

Figure 1.1. IoT Reference Model [1], [2], [3].

1.1.2. Fog Computing

The IoT is generating an unprecedented volume and variety of data at unprecedented speeds. To keep up with IoT data sources, analysis of IoT data must be very rapid. Handling the volume, variety, and velocity of IoT data requires using the computing model of fog computing (FC) [7], implemented via fog nodes, illustrated in Figure 1.2.

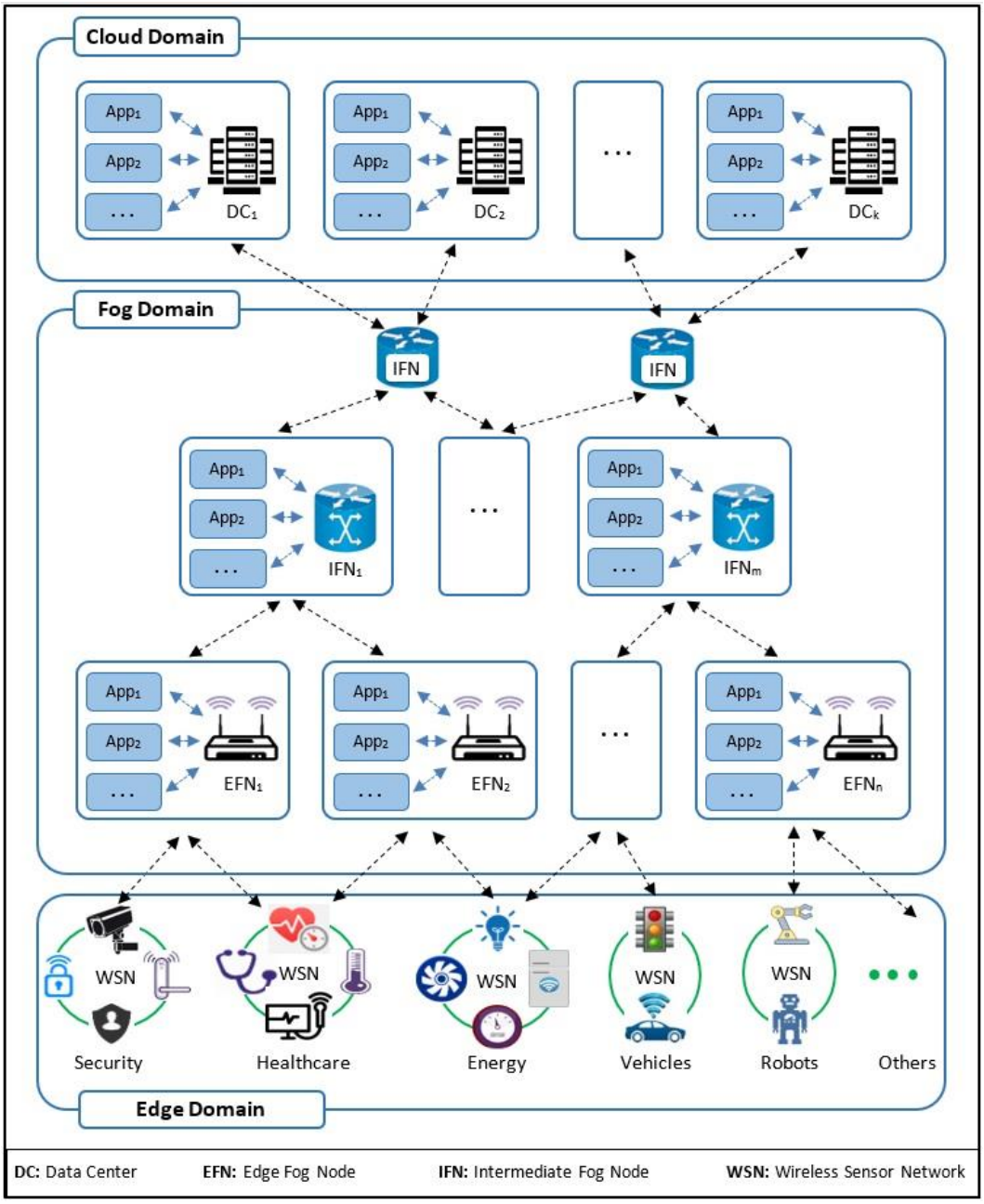

Figure 1.2. Fog Computing Architecture [7].

Fog nodes can be deployed anywhere at the edge where there is a network connection: on a factory floor, on top of a power pole, alongside a railway track, in a vehicle, or on an oil rig. Examples include industrial controllers, switches, routers, embedded servers, and video surveillance cameras. Fog node has several functionalities [7], including: (1) Receiving feeds from

IoT devices, some using real-time protocols; (2) analyzing the most time-sensitive real-time data at the edge of the IoT network, as close to their sources as possible; (3) running real-time applications on hardware devices that virtualize remote cloud services for these applications; (4) providing temporary data storage; and (5) sending either raw delay-tolerant[1] data or pre-processed delay-tolerant data (e.g., aggregations or periodic data summaries) to the cloud for non-real-time applications, including historical analyses and long-term or permanent storage.

The most interesting aspect of a fog node is that it is a system that can, on the one hand, control a specific set of edge devices, while on the other hand, access the cloud. Figure 1.2 illustrates the FC architecture within the IoT environment (incl. cloud computing) and shows our design for fog nodes as well as their deployment in the IoT.

There are two types of fog nodes—based on their locations and functionality [8], as shown in Figure 1.2 above: (a) *edge fog nodes (EFN)* are located at the edges of IoT networks, closest to data sources (IoT devices), and designed to perform fog computing for IoT data collected within local sensing domains. As much pre-processing (incl. aggregation and analysis) of IoT data as possible should be done by the EFN that collect IoT data. If processing (e.g., aggregation) requires data from, say, EFN_1, EFN_2, and EFN_3, data from EFN_2 and EFN_3 can be sent to EFN_1 (assuming, e.g., that this minimizes the amount of transferred data), and processed there; and (b) *intermediate fog nodes (IFN)* located between EFNs and the cloud to forward EFN data to the cloud, or forward cloud data to EFNs (e.g., for actuators). IFNs also can perform pre-processing (incl. aggregation and analysis) of IoT data collected from EFNs.

[1] We use *delay-tolerant* as a synonym for not time-sensitive.

We distinguish two kinds of IoT applications from the point of view of a given fog node *N*. First, an application running within *N* is its *local application*. In Figure 1.2, App_1, App_2, … running within EFN_1 are local applications for EFN_1. Second, any application running on any node other than *N* is a *remote application* for *N*. In Fig. 1.2, App_1, App_2, … within EFN_n as well as App_1, App_2, … within Cloud, DC_2 are remote applications for EFN_1. A fog node consists of physical and virtual components that work together to perform fog computing. A fog node structure [8], illustrated in Figure 1.3, includes: (a) *hardware* consists of all the physical parts for processing, storage, and communication; (b) *operating system (OS)* includes all the software components required to support Fog middleware, and applications (either via VM hypervisor or API Manager); (c) *Middleware* includes fog area network (FAN), that handles software-defined networks (SDNs) [9], network function virtualization (NFV) [10], and wireless sensor networks (WSNs) [11], IT and data abstraction [12], services support, and applications support; and (d) *VM hypervisor* with *VMs* for real-time applications; and (e) *API manager* and *APIs*.

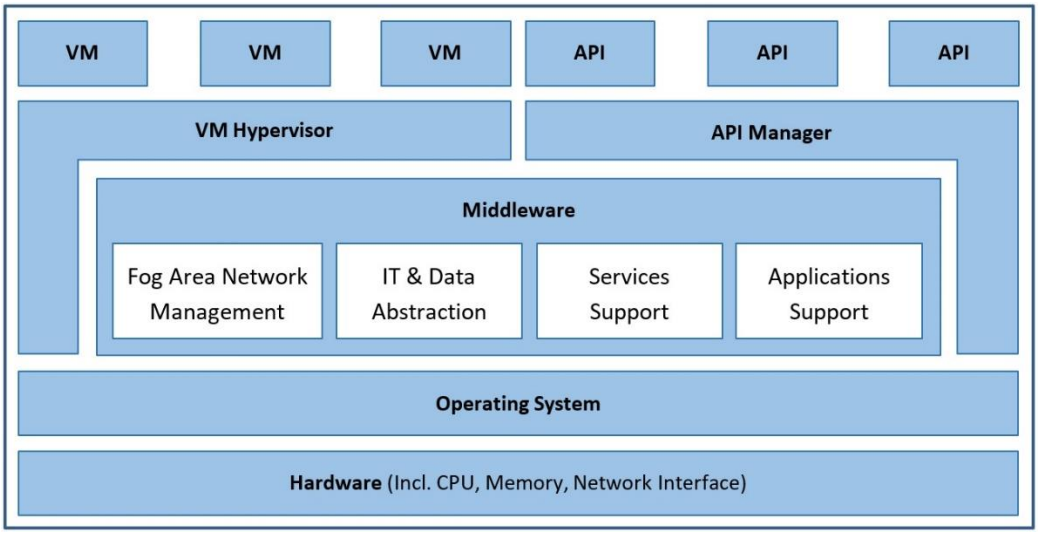

Figure 1.3. Fog Node Structure [8].

1.1.3. IoT Data Privacy

Data privacy is a critical issue for the IoT community. First, due to its myriad sensors and other devices operating in private homes, businesses, and public environments, IoT can collect huge amounts of data on individual entities, raising the threats of tracking and surveillance for these entities. Further, a collection of these data can occur without the knowledge or informed consent of these entities. This raises the issue of a lack of an entity's control over sensitive data describing the entity, which should be owned (hence controlled) by this entity. Admittedly, data collected by IoT devices in most cases (when used legitimately) will benefit the device's owner in terms of the provided services. However, even in these cases, the same data can, in addition, be used in ways that violate the privacy of the device's owner, and instead of benefiting him, benefit the device's manufacturer, service providers, or other parties. This includes cases when sensitive data are used for reasons other than the stated data collection purposes [13]. Accordingly, data privacy should be protected by ensuring that data owners (or guardians) control which of their sensitive data are being collected, who is collecting such data, and when this is happening. Further, collected sensitive data should be used only in ways accepted by their owners, for supporting authorized services. Finally, sensitive data should be destroyed when no longer needed for the owner-approved goals.

1.1.4. Sensitive Data

Data considered sensitive based on certain factors [14], include being inherently sensitive, coming from a sensitive source, declared sensitive by their owner, being in a sensitive context, being a part of sensitive data, or being sensitive in relation to other sensitive data. Based on the level of sensitivity, data can be classified into [15]: (a) *restricted* data are the most sensitive data

that cause a great risk if compromised. Access is on a need-to-know basis only; (b) *confidential or Private* data are moderately sensitive data that cause a moderate risk if compromised. Access is subject to privacy policies defined by data owners, and (c) *public* data are non-sensitive data that cause little or no risk if accessed. Access is loosely controlled or not controlled.

1.1.5. Privacy Policies

Privacy policies are the rules that determine the authorized operations on given data, such as allowing or denying read, write (update), or delete operations [14]. Privacy policies are typically defined by data owners or defined as a form of agreement between clients and service providers. In some cases, privacy policies can be generated automatically. In other cases, a system can assist a data owner in defining privacy policies for his/its data. There are four characteristics describing privacy policies [16]:

1) *Purpose* specifies the reason or intention for using data. Privacy of a data item is breached if it is used for a purpose not specified in its privacy policy.
2) *Visibility* refers to the categories of data users who can access data. Typically, data owners select visibility for their data at one of these levels:
3) *Granularity* refers to the smallest portion (granule) of data that can be accessed. For example, a granularity of access for one database could be an individual field, for the second database—a whole record, and for the third database—the whole file.
4) *Obligations* are a set of actions that must be performed by an entity accessing data either before or during the access. For example, a service provider must obtain a customer's permission before accessing her private data, and this access must be based on the need-to-know principle.

1.1.6. Data Ownership

The following are different categories of data ownership:

1) *Data Owner (DO)*: For given data, data owners are the individuals or entities being described by these data, or the individuals or entities that own the devices that created the data describing them [14].

2) *Data Guardian (DG)*: For given data, data guardians are the individuals or entities to whom a data owner delegates his data ownership rights (incl. specification of data privacy policies). For example, a system administrator or a doctor may become a guardian for a patient's data [17].

3) *Data User (DU)*: For given data, data users are the individuals or entities who handle the usage of these data. Data usage includes processing, analyzing, storage, receiving notifications, etc. DU must use data for legitimate purposes based on the need-to-know principle [18].

1.1.7. Selected IoT Privacy Threats

The diversity of IoT applications serving diverse entities, including end users, leads to many privacy vulnerabilities. The most common data privacy threats[2] in IoT are as follows:

1) *Identifiability threat*—when an adversary can identify an identity from specific data [19].

2) *Linkability threat*—when sensitive data can link "separate" data to an individual identity [20].

[2] PEFM provides controls for all identified privacy threats, using appropriate policy sets (including privacy rules).

3) *Unauthorized data disclosure threat*—when sensitive data are exposed to unauthorized parties [14].

4) *Excessive data disclosure threat*—when more sensitive data than needed are exposed to *authorized* parties [21].

5) *Profiling threat*—when it is possible to infer the interests and habits of individuals or entities from their data and metadata [19].

1.1.8. Privacy Controls in IoT

Privacy can be controlled by satisfying the privacy services [22] which—working in a certain combination—can be called upon to assure privacy via cooperative implementation of a privacy-preserving solution (which uses them).

We recognize the following set of privacy services [20], [22], [23]:

1) *Confidentiality* is the control of protecting sensitive data from unauthorized disclosures or unauthorized disseminations.

2) *Appropriateness* is the control of assuring that the collection, processing, and retention of sensitive data must be done only for legitimate purposes and only on an as-needed basis.

3) *Anonymity* is the control of hiding the real identity of data owner during data processing, disclosures, and disseminations. Or it can be defined as hiding the identity of an entity described by a dataset.

4) *Untraceability* is the control of preventing an adversary from finding out that a given set of actions was performed by the same subject.

5) *Unlinkability* is the control of hiding information about relationships among items (e.g., entities, messages, actions, etc.).

6) *Unobservability* is the control of hiding data items so that an adversary cannot see the item of interest.

7) *Notification, rights, & consent* are the controls of assuring the rights for data owners to be notified about the sensitive data collected by them and to give consent for their use.

1.1.9. Active Data Bundle

An *Active Data Bundle (ADB)* is a software construct that encapsulates sensitive data (to be protected) with metadata and a virtual machine that controls data and metadata [24], as shown in Figure 1.4. The main functions of an ADB are (a) protecting its sensitive data from unauthorized disclosure and unauthorized dissemination by outsiders, and (b) limiting an authorized user (an insider) from going beyond the need-to-know boundaries when accessing sensitive data. The ADB scheme transforms data from passive entities (which must be manipulated by external active entities) to active entities (which are self-contained in terms of operations that they can perform themselves).

Each ADB includes the following components:

1) *Sensitive data* can include any digital content such as documents, code, images, audio, video, etc. Sensitive data lifecycle includes creating data (by an owner or a *creator* acting on his behalf), disseminating data, disclosing data (full, partial, or null), and destroying data (at the end of data lifetime).

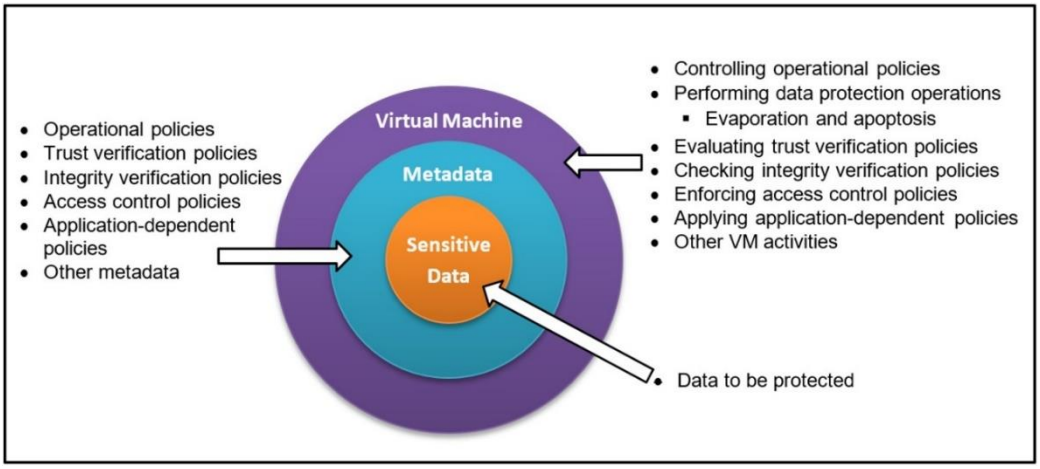

Figure 1.4. Active Data Bundle [17], [24].

2) *Metadata* include all privacy policies required to control ADB behavior and operations (enforced by the VM). Metadata include, among others, the following *categories of privacy policies*:

   a) *Operational Policies*—control ADB behavior during its lifecycle from creation to destruction.

   b) *Trust Verification Policies*—verify that the trust level of the host to be visited is sufficiently high to allow their ADB to visit the host.

   c) *Integrity Verification Policies*—verify the integrity of ADB components (data, metadata including policies, and the VM including mobile code).

   d) *Access Control Policies*—control data disseminations and data disclosures. The former requires specifying who can disseminate the ADB that contains them, and under what conditions. The latter provide conditions upon which their ADB may disclose specific data to specific hosts.

e) *Application-Dependent Policies*—specify different policies for different applications; this includes the case when the *same* data are used by *different* applications.

3) *Virtual Machine (VM)* is a program designed to enforce all policies and rules defined within ADB metadata (and, possibly, within the VM code as well).

A VM of an ADB makes its ADB active by executing the following main functions:

1) Controlling operational policies that manage the ADB behavior during its lifecycle.
2) Performing the following data-protecting ADB's operations [14]:
   a) *Evaporation*—the process of destroying the part of ADB data that the host is not allowed to access (due to the insufficient trust level of the host, and the conditions specified by ADB policies).
   b) *Apoptosis*—the process of complete self-destruction of an ADB when it's VM realizes that privacy of the ADB is about to be compromised (e.g., based on the trust level of the host to be visited or due to ADB policies). It is also performed when an ADB has disclosed data to a host and has no more hosts to visit.
3) Evaluating trust verification policies to assure ADB protection from unauthorized receiving hosts. It involves checking the trust value of the visited host and comparing with ADB's *trust threshold*.[3] If the trust value is lower than the threshold, the VM performs apoptosis. Otherwise, the VM authorizes disclosing data to the host. (More specifically, the VM authorizes disclosing data to one or more applications running on the host, at the disclosure levels appropriate for each of these applications.)

[3] A *trust threshold* shows the minimum trust value that an ADB requires for a host to disclose any data at the host (to any application at the host).

4) Checking integrity verification policies to assure ADB integrity. This involves the VM calculating the hash value for the ADB at the visited host. If it is different than the hash value ATT-HV (calculated during ADB creation and stored within ADB metadata), the VM apoptosizes the ADB. Otherwise, the VM proceeds to the following step.

5) Enforcing access control policies to determine the extent of data disclosure. The VM evaluates policies P for data delivered to Host H, and—as determined by this evaluation—performs an appropriate data disclosure:

   a) *Full data disclosure*—permitting disclosure of all ADB data.

   b) *Partial data disclosure*—permitting disclosure of a part of ADB data.

   c) *Null data disclosure*—preventing disclosure of any of ADB data.

6) Applying application-dependent policies to facilitate delivering data to a requesting application.

1.2. Motivation

In this subsection, we outline the motivation for protecting sensitive IoT data, and for using policy enforcement as an approach for protecting these data.

1.2.1. Motivation for Protecting Sensitive IoT Data

Data generated from IoT objects are the valued product for adversaries, as they can contain sensitive information—such as personally identifiable information (PII), payment card information or protected health information. As the number of IoT-connected devices and sensors

increases, the amount of sensitive data collected at the IoT edge and moving into the backend in the data centers (including clouds) is growing exponentially [25].

A survey conducted among IT professionals indicates data privacy as the second most important challenge for adoption of IoT [26]. This motivates us to investigate the issue of data privacy in IoT, and to propose a solution.

The rate of sensitive data use by IoT devices is 33% [27]. Sensitive data are a significant portion of data used by big data applications, as well as different types of cloud services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) (which are considered main IoT data services) [28]. As high as these sensitive data rates are, they are expected to grow in the future—due to the increase of IoT data sources, services, and applications. This makes privacy concerns one of the most critical issues for IoT. Many organizations have strong concerns about the privacy of their IoT data; 33% of IT professionals are concerned with protecting sensitive data generated by IoT, the highest rate among IoT privacy concerns [27].

1.2.2. Motivation for Policy Enforcement

Among the many mechanisms and approaches proposed in the literature, we choose *privacy policy enforcement* as a major privacy protection approach to be pursued in our research. The main idea for this approach can be outlined as follows: (a) associating data to be protected with a set of privacy policies that determines usage rules for these data; (b) for each data use request, evaluating applicable policies for the requested data, and determining if and in which way these data may be used; and (c) using a policy enforcement mechanism to enforce the results of policy evaluation.

The reasons for selecting the privacy policy enforcement approach include:

1) *Flexibility*: Since data to be protected come from large networks of heterogeneous IoT devices, the corresponding privacy protection mechanism must be flexible enough to meet demands for a wide variety of data with diverse protection requirements. Privacy policies can be set regarding data types and their degree of sensitivity and can determine who can access which sensitive data and at what level of access. Flexibility in defining policies will assure flexibility in data protection, suitable for different situations.

2) *Data owner control*: There is a lack of control over sensitive data shared in IoT by their owners with other entities. Our approach to addressing this issue relies on using ADBs, which encapsulate data with their privacy policies and their policy enforcement code in an inseparable way. This means that wherever data are accessed, their privacy policies defined by their owner will be enforced. Hence, the data owner's control is achieved even if the data leave their owner's domain.

3) *Lifecycle data protection:* Data privacy breaches can happen at any stage of the data lifecycle, so assuring data protection throughout their entire lifecycle is required. The proposed Local Policy Enforcement Module (LPEM) module protects data during collection and local fog processing stages, while the proposed Remote Policy Enforcement Module (RPEM) module protects data during dissemination for remote processing outside of the fog node by which the data were collected.

1.2.3. Motivation for Fog Computing

Fog computing (FC) has many advantages that motivated us to adopt it in our solution [7]. In this subsection, we discuss a selected set of advantages toward enhancing privacy control in the IoT environments as follows:

1) *Better Privacy Control*: Implementing privacy control in FC is significantly decrease the privacy risk associated with sending unprotected data to remote nodes.

2) *Better Security:* FC enables system administrators to monitor and control data processing and analysis in their local domain that has low-security risks instead of sending data over the networks that have high-security risks.

3) *Data Selectivity*: Most of the data generated at the edge are processed by fog nodes. Only selected data are sent to the cloud for more processing and long-term storage. This decreases the privacy risks associated with sending all data to the cloud.

4) *Efficiency*: FC conserves network bandwidth by processing most of the data at the edge instead of sending them to the cloud for processing. The FC also minimizes latency for data processing and transmission.

5) *Fast Responses for Time-Sensitive IoT Applications*: FC enables local decision making and hence fast response for real-time applications that require an immediate response. In terms of privacy, lightweight privacy mechanisms are good enough for such applications since data are consumed at the edge with low risk.

6) *Geographical Distribution:* Data and their applications are widely distributed so instead of acting on a huge amount of data in the cloud, FC enables acting on data in a distributed manner. This helps in applying privacy mechanisms to distributed data instead of on a large amount of data that require high computation cost for privacy.

7) *Local Processing of Data*: FC assures acting on data as close as possible to the data owner's local domains, which are typically trusted by those data owners. This decreases the privacy risks associated with acting on data in remote (untrusted) cloud domains.

1.3. Problem Statement

The problem to be addressed by the proposed research can be stated as follows:

Sensitive data in the Internet of Things (IoT) are vulnerable to privacy violations, which include attacks by outsiders who are not authorized to access these data; as well as accidental leaks or deliberate attacks by insiders who exceed their need-to-know boundaries when accessing these data. Furthermore, there is a lack of data owners' control over their sensitive data when these data leave their owner's domain and are disseminated to other IoT entities.

1.4. Research Hypothesis

The research problem specified in the problem statement above can be addressed by proposing a solution outlined by the following hypothesis:

Privacy of sensitive IoT data can be protected by using a policy enforcement module within fog nodes. We refer to this module as Policy Enforcement Fog Module (PEFM). For local (real-time) IoT applications PEFM uses Local Policy Enforcement Module (LPEM) that enforces privacy policies directly, and for remote (non-real-time) IoT applications PEFM uses Remote Policy Enforcement Module (RPEM) that enforces privacy policies indirectly via Active Data Bundles.

1.5. Research Scope

There are a few limitations in our research that are beyond the scope of this dissertation and must be left for future investigation. We limit our scope as follows:

1) Data in IoT might be public and private (sensitive). We consider only sensitive data. Public IoT data are outside of the scope of this research.
2) We rely on the data owner to identify which data are sensitive, so identifying sensitive data is outside of the scope of this research.
3) We focus on protecting sensitive IoT data. Protecting other aspects of privacy for IoT entities is outside of the scope of this research.
4) We investigate the privacy of data. Security is outside of the scope of this research. However, some security issues, intertwined with privacy issues [29], are addressed by our research.
5) We limit our investigation to the selected privacy threats (as discussed in Subsection 1.1.7). Other threats are outside of the scope of this research.

6) We limit our investigation to fog computing rather than any similar terminologies—including cloudlets [30], Mobile Edge Computing [31], Mobile Cloud Computing [32], and so on.

1.6. Main Contributions

The main contributions of this dissertation include the following:

1) We propose the *policy enforcement fog module* for protecting sensitive IoT data throughout their lifecycle. The proposed module protects sensitive IoT data from unauthorized disclosures and limits authorized access to these data based on need-to-know only.
2) By the proposed module, we enhance fog computing by adding to it a mechanism for supporting data privacy.
3) By the proposed solution, we extend privacy protection for fog data to non-fog data using the power of *active data bundles*. PEFM can serve both local (real-time) IoT applications via *local policy enforcement module* and remote (non-real-time) IoT applications via *remote policy enforcement module*.
4) We develop a comprehensive simulation of the PEFM scheme. The simulation demonstrates the feasibility of the module by being able to effectively protect sensitive IoT data with reasonable costs in terms of system latency and system throughput.
5) We present a sample application of PEFM module for a home intelligent surveillance system.

1.7. Dissertation Organization

The rest of the dissertation is organized as the following. Chapter 2 explores related research on data privacy in IoT. Chapter 3 gives an overview of PEFM, our proposed solution, and outlines a high-level design of PEFM including its structure (components) and process flow (operation). Chapter 4 presents a proof-of-concept case study. Chapter 5 describes the simulation design for the PEFM system and outlines the setup of a simulation for the case study. Chapter 6 presents a performance evaluation of the PEFM system. Chapter 7 summarizes the dissertation, concludes it, and outlines the future work.

CHAPTER 2

LITERATURE REVIEW

To address privacy concerns in IoT, many solutions have been proposed in the literature during the past few years. In this chapter, we classify the related work based on the technique or the mechanism used for assuring privacy. Our classification includes four major privacy solution categories, namely encryption-based solutions, identity management-based solutions, policy enforcement-based solutions, and self-protecting solutions. For each category, we discuss different state-of-the-art solutions; we first compare them with our work from a general perspective and then specifically in terms of the privacy goals of the proposed PEFM solution. We discuss solutions which adopt Active Data Bundle (ADB) scheme [24] as a mechanism for protecting sensitive data and distinguish our work from them.

2.1. Encryption-Based Privacy Solutions

This subsection outlines the privacy solutions in the literature that use encryption as a basis for assuring privacy. Most of these solutions focus on protecting the confidentiality of data by encrypting them.

2.1.1. Public-Key Encryption

Public-Key Encryption (PKE), for example, Diffie-Hellman Encryption, is an asymmetric cryptographic system that uses a pair of keys: public key (PK) and secret key (SK). The PK may

be disseminated widely so everyone can know it. The SC must be known only by the intended recipient of data to decrypt and be authenticated to receive the encrypted data. The strong point of PKE in terms of privacy is each receiver must have a unique SC. This assures two-party authentication, confidentiality, and integrity. A security protocol for sensor network (SPINS) [33] uses PKE to assure secure communication between sensor nodes and the base station. Since PKE requires computation power, the SPINS use the base station to set up the keys. The SPINS protocol is best suited for a small-size network with node-to-base communication pattern [34]. This makes it limited to secure local networks such as Wireless Sensor Networks (WSNs) rather than securing IoT network beyond the WSNs.

2.1.2. Attribute-Based Encryption

The concept of Attribute-Based Encryption (ABE) is an extension or enhancement of identity-based encryption (IBE) [35], which preserves the privacy of entities by providing a fuzzy identity based on combining an entity's identity and a series of its descriptive attributes. Messages are encrypted using a set of attributes that describe the intended receivers.

There are two major types of ABE schemes: Key-Policy ABE (KP-ABE) [36], [37], where only users that hold a specific set of the attributes can decrypt the messages; and Ciphertext-Policy ABE (CP-ABE) [38], [39], where attributes are used to describe an entity's credentials, and a data owner encrypting his data provides a policy determining who can decrypt them. CP-ABE controls access to encrypted data to assure data confidentiality. Wang et al. [40] consider using KP-ABE and CP-ABE for IoT data privacy and found that KP-ABE and CP-ABE are crucial for achieving data privacy in IoT. Since ABE schemes provide public key encryption, they enable fine-grained access control, scalable key management, and flexible data distribution. These properties can be

useful in some IoT cloud applications [41]. However, ABE schemes are usually computationally expensive as they use bilinear pairings, making them less suitable for resource-constrained IoT applications. Recently, a lightweight attribute-based encryption scheme (which does not use pairings and relies on elliptic curve cryptography) has been proposed for the IoT [42].

2.1.3. Homomorphic Encryption

Homomorphic encryption allows data owners to encrypt sensitive data and enables services to process encrypted data without their decryption. There are two basic types of homomorphic encryption schemes: Partially Homomorphic Encryption (PHE) and Full Homomorphic Encryption (FHE) [43]. Since FHE schemes are very computationally and memory demanding, many PHE schemes have been proposed [44], [45].

Homomorphic encryption can be a part of a secure multi-party computation [46]. Unfortunately, in practice, it is limited to a small set of simple operations on data (including addition and multiplication operations) [29]. Sun et al. [47] propose a privacy protection solution for IoT services based on the multiplication homomorphism. Example applications include smart grids [48], WSN [49], IoT-based healthcare monitoring [50], and IoT data collection services [51].

2.1.4. Point-to-Point Encryption

Point-to-Point Encryption (P2PE) is a standard for encryption established by the Payment Card Industry (PCI) Security Standards Council [52]. Encryption is performed between two designated and independently validated devices. The source device represents the point of encryption, and the destination device represents the point of decryption. Encrypted data is subsequently sent by the source device as ciphertext (unreadable for intermediate devices) to be

decrypted by the destination device in the network. This means the data is secured and cannot be decrypted by unauthorized users who might steal it during the transaction. Hence, P2PE can assure data privacy by assuring the confidentiality of data.

P2PE aims to provide a payment security solution by converting sensitive payment data into an indecipherable code at the time the card is swiped during a business transaction. P2PE provides a high level of confidentiality of data. However, it is computationally demanding due to the use of comprehensive cryptographic and key management systems as well as the need for devices validations. P2PE is limited to business IoT applications and cannot assure fine-grained access control for sensitive data like PEFM aims to do.

2.2. Identity Management-Based Privacy Solutions

This subsection outlines the privacy solutions in the literature that are using identity management as a mechanism for assuring privacy. Most of these solutions focus on protecting the identity of the source of data or the destination as well.

2.2.1. K-Anonymity

One of the most known privacy protection approaches is k-anonymity [53], which protects against identity disclosure by guaranteeing that each released "piece of information" relates to at least k records (hides in the crowd of k records). However, k-anonymity does not sufficiently protect against attribute disclosure [54]. Huang et al. [55] propose a solution using a context-aware k-anonymity. The proposed approach manipulates a record's identifier so that a record cannot be distinguished from k-1 records. Gope and Hwang [56] propose an anonymous authentication scheme designed to assure privacy in a distributed IoT system. The scheme assures sensor

anonymity, untraceability, resistance to replay attacks and cloning attacks, as well as mutual authentication. However, it does not guarantee data authenticity, and such privacy properties like non-repudiation, revocation, etc.

2.2.2. Pseudonymity

Pseudonymization replaces a user's real identity with a pseudonym [20]. It is an efficient mechanism preventing identification and tracking in IoT applications. Pseudonymization is a form of de-identification (different than de-identification used for anonymization) for applications that might require re-identification. In contrast to a user of anonymized data, a user of pseudonymized data can re-identify them easily (when provided with the pseudonym-to-identifier mapping by the party that pseudonymized the data). This scheme is limited to protect a user's identity rather than protecting user's data that includes personally identifiable information (PII), which is our approach.

2.2.3. Attribute-Based Signature

Attribute-Based Signature (ABS) allows users to generate signatures based on their attributes, which can be verified for satisfying data access policies without leaking sensitive data [57]. Once a user requests data or services; he must generate a signature using his attributes. The ABS scheme assures that signers remain anonymous (all users are indistinguishable, hiding in the crowd), and assures that only a user with valid attributes can receive data or services from the IoT infrastructure. Usually, the ABS schemes are based on attribute-based credentials that are used in attribute-based user authentication systems, such as Idemix [58], U-Prove [59] and HM [60]. This approach is limited to assure de-identification of IoT users, so it does nothing for IoT data

transmitted among IoT services and users. Therefore, this approach is not appropriate to address our privacy issue to protecting sensitive IoT data.

2.2.4. Group Signatures

Group signatures are a generalization of membership authentication schemes, in which a group member can verify whether he belongs to a particular group, without revealing his identity. The receiver of a signed message can check whether it has a valid signature of that group but cannot discover which member of the group signed it [61]. Many researchers use group signatures for Mobile Ad-hoc NETworks (MANETs) [62] and Vehicular Ad-hoc NETworks (VANETs) [63] in order to assure privacy and anonymity of senders for broadcasts in MANETs and VANETs (which can be subsystems within IoT). Group signature schemes are not suitable for resource-constrained devices because they use computationally expensive operations, such as exponentiation and bilinear pairing.

2.2.5. Ring Signatures

Ring signatures (RS) protect privacy (identity) of a data sender by assuring that a message can be delivered only to a member of the predefined ring of users, who knows the ring signature. Signcryption [64] is a high-performance cryptographic primitive that uses a single step for both digital signatures and public key encryption, at a cost significantly lower than those of signature-then-encryption or encryption-then-signature approaches. It simultaneously achieves confidentiality, authentication, integrity, and non-repudiation without the need for certificates. Due to cost reduction, signcryption is very suitable for resource-constrained devices (such as sensors). In contrast to signcryption, the “unsingcryption” algorithm needs a powerful device (such as a

server) to calculate n point multiplications, two pairing operations, and a few less expensive operations (hashes, additions, etc.). Li et al. [65] propose a heterogeneous "signcryption" ring for securing IoT data transmissions from sensors to servers. RS is suitable for IoT applications. However, it is dedicated to protecting the identity of a data sender in a ring of users rather than protecting user's data from security or privacy violations, which is our focus.

2.3. Policy Enforcement-Based Privacy Solutions

This subsection outlines the privacy solutions in the literature that are based on policy enforcement as a mechanism for assuring data privacy. These solutions include defining privacy policies for data to be protected then evaluate these policies for every access to the data and enforce the policy enforcement decision.

2.3.1. Mandatory Enforcement of Privacy Policies

Kargl et al. [66] introduce the concept of mandatory enforcement of privacy policies. This concept empowers data owners to tightly couple their data with privacy policies and rely on the system to impose such policies onto any data processors. They exemplify their approach by describing the PRECIOSA Privacy-enforcing Runtime Architecture for Cooperative Intelligent Transportation Systems (cITS). PRECIOSA uses a chain of data protection mechanisms: (a) policies describing the conditions under which data controllers and processors may use the data; the policies are stated explicitly and are tightly coupled with the data they govern; (b) Mandatory Privacy Control (MPC) enforcing privacy policies whenever personal data is accessed; and (c) Mandatory Integrity Protection (MIP) establishing trust in remote systems, and ensuring that only both proper recipient and properly functioning MPC are able to access the data. This approach is

different from our solution by relying on the centralized trusted third party (MIP) to enforce privacy policies; in contrast, our solution relies on fog nodes working in the data owner's local trusted domain for protecting real-time IoT data, and relies on the ADBs created in the fog nodes to enforce privacy policies for non-real-time data disseminated to remote nodes for higher-level processing and long-term storage. Fog computing paradigm, used by us, proposed to serve real-time processing for IoT data without the need to send these data to the TTP. Using TTP increases network traffic and adds extra computations for authentications as well as the known bottleneck problem of TTP. This derives us to avoid using TTP in our solution, again, we use fog computing instead.

2.3.2. Policy-Based Management

Dsouza et al. [67] propose policy-based management of resources in fog computing. They build a policy module that manages interoperability among different user-requested fog resources and supports secure collaboration. Each time when a service is requested by an IoT user, policies are enforced on that service. This approach differs from our approach by applying policy enforcement for controlling the use of services. In contrast, we apply policy enforcement for controlling access to sensitive IoT data. Further, they consider policies that manage interoperability and secure collaborations of users, while we consider policies as data privacy controls.

2.3.3. Cloudlet-Based Privacy Mediator

Cloudlets are small locally-administered data centers run within a user's own trusted domain. Davies et al. [68] propose a privacy mediator, which is a cloudlet-based (hence locally-

controlled) software component. It runs for a single raw sensor-data stream and enforces privacy policies for a single sensor. This means that typically there are several privacy mediators in the same administrative domain (one for every raw sensor-data stream). Privacy policies are defined by the owner of data being collected by sensors and dynamically enforced within the user' domain. This approach has many parallels with our approach, such as enforcing privacy policies on raw data streams from IoT sensors and enforcing policies closest to data sources. However, it differs from our approach in a few respects. First, it limits the deployment of the mediator to cloudlets, while we consider deployment of the PEFM module for fog computing that is more general than cloudlets. Second, it limits the protection to a user' trusted domain, while—via utilization of ADBs—we extend privacy protection to untrusted domains, including public clouds and other services.

2.4. Self-Protecting Based Privacy Solutions

This subsection outlines the privacy solutions in the literature that are self-protecting data approaches. Most of these solutions rely on the idea of attaching data with a mechanism that assures no access can be done for these data unless the privacy protection is done first.

2.4.1. DigiBox

Sibert et al. [69] propose the DigiBox self-protecting container technology for e-commerce. The DigiBox provides data containers designed to protect private data during transmission through network nodes and assure rights protection. DigiBox is cryptographically safeguarded to secure private data transmission. It also controls the enforcement of data rules to ensure data confidentiality. DigiBox provides a secure container for packaging sensitive data in a way that the

data cannot be accessed except as provided in the rules associated with the data. Therefore, DigiBox can be used to assure the confidentiality of sensitive IoT data. However, it doesn't guarantee data lifecycle protection like PEFM does.

2.4.2. Agent-Based Informed Consent

Neisse et al. [70] use the notion of informed consent, which denotes getting permission for accessing personal data from data subjects, that is, the users whom these particular data describe (note that the model, using the terms "personal" and "users," was created for human entities). They propose an agent-based framework for informed consent in IoT. They define informed consent policy rules and specify enforcement policy rule templates called profiles, which include privacy policies. To preserve privacy, subjects in an IoT system must be fully informed about the informed consent process, and the use of their personal data. Each data subject can select a profile (with privacy policies) from a set of predefined profiles; hence, the subject does not need to define his privacy policy from scratch. The proposed framework controls the flow of data collected and transmitted by from IoT devices to service providers. It includes a user-centric mechanism to deploy and manage privacy policies (including their evaluation and enforcement) for local smart spaces [71]. In contrast, we use data-centric mechanisms LPEM and RPEM, the latter employing ADBs. We use a data-centric approach, rather than user-centric approach because we believe that protecting data, collected about users, assures protecting the privacy of those users. While protecting users' privacy in the mean of identity or location does not guarantee the protection of their data.

2.4.3. Personal Digital Rights Management

Personal Digital Rights Management (PDRM) is a self-protecting data approach enabling individuals to protect their privacy rights for their sensitive data [72]. PDRM attaches to data a software construct named a detector, generated using an Artificial Immune System (AIS) technique. The detector assesses the use of the data to which it is attached and denies data access if it recognizes an unusual access pattern. PRRM is limited to allow or deny decisions only and doesn't support partial data disclosures.

2.5. Comparison of Related Work with PEFM

Table 2.1 compares the proposed solution PEFM with the related work in the literature in terms of a set of privacy goals achieved by the PEFM solution.

2.6. Protecting Sensitive Data with Active Data Bundles

Many researchers investigated using Active Data Bundle (ADB), or a.k.a Active Bundle (AB), as a promising mechanism for protecting sensitive data in diverse systems including IoT, healthcare, VANETs, and cloud. In this section, we discuss selected work that uses ADB for data privacy.

Table 2.1. A comparison of the related work with PEFM in terms of privacy goals.

Category	Privacy Goal							
	Increase User Control	Selective Data Disclosure	Fine-Grained Access Control	Enable Data Minimization	Enable Data Destruction	Enable Unobservability	Make Data Active	Entire Lifecycle Protection
PEFM	√	√	√	√	√	√	√	√
PKE	√	–	–	–	–	√	–	–
ABE	√	–	–	–	–	√	–	–
Homomorphic Encryption	√	–	–	–	–	√	√	√
P2PE	√	–	–	–	–	√	–	–
K-Anonymity	√	–	–	–	–	√	–	√
Pseudonymity	√	–	–	–	–	√	–	–
ABS	√	–	–	–	–	√	–	–
Group Signatures	√	–	–	–	–	√	–	–
Ring Signatures	√	–	–	–	–	√	–	–
PRECIOSA	√	√	√	√	–	√	–	–
Policy-Based Management	–	–	√	–	–	√	–	–
Privacy Preference Model	√	–	√	√	–	√	–	–
Cloudlet Based Privacy Mediator	–	–	√	√	–	√	–	–
DigiBox	√	√	√	√	–	√	–	–
Agent-Based Informed Consent	√	–	√	–	–	–	√	√
PDRM	√	–	√	–	–	–	√	√

Ben Othmane and Lilien [24] first proposed ADB for protecting sensitive data from their disclosures to unauthorized parties and from unauthorized dissemination (even if started by an authorized party). The ADB solution protects private or confidential data throughout their entire lifecycle, from creation through dissemination to partial or destruction. The ADB, as described earlier, is a software construct that encapsulates raw data (to be protected) with metadata and a

virtual machine that controls data and metadata. It uses two major protection operations: evaporation and apoptosis. Evaporation is the process of destroying the part of ADB data that the host is not allowed to access. Apoptosis is the process of complete self-destruction of an ADB when it's VM realizes that privacy of the ADB is about to be compromised due to ADB policies.

Angin et al. [73] propose an entity-centric approach based on AB for privacy and identity management in Cloud computing. They use AB to protect personally identifiable information (PII) of the cloud's users. Whenever a user uses a cloud service, the cloud's service provider requires PII to authenticate that user. With AB, a set of privacy policies, within AB's metadata, used to protect user's PII from privacy violation by the service provider. These privacy policies assure anonymous identification of cloud users by service providers; and hence, ensure the privacy of these users.

Salih and Lilien [74] propose using AB to provide protection for the patients' Electronic Medical Record (EMR) during the entire EMR lifetime, including its dissemination among different healthcare provider locations. First, the EMR owners using ABs to prevent unauthorized EMR access by visited hosts with insufficient trust levels. Second, ABs provide tools that are protecting data owners' (or guardians') privacy rights for arbitrary EMR fragments (down to the single record level). Third, an AB protects a patient's EMR even if the EMR is owned by different owners with privacy policies of differing strength. Even if an attacker tries to exploit the laxness of some owners' privacy policies, he still faces the AB's VM that enforces the required level of privacy via AB's privacy policies.

Ranchal and Bhargava [75] propose an approach for secure data dissemination among Product Lifecycle Management (PLM) systems using the AB scheme. The proposed approach enables Enterprises to securely share information with their partners during the PLM steps and

protects it throughout the entire product lifecycle. They use AB to protect the highly sensitive information such as trade secrets, intellectual property, and private organizational details during their dissemination across large enterprise systems. Each organization defines its privacy policies for data that need to be shared with other organization, and the AB assures no data are delivered to any destination without enforcing the owner's (source organization) privacy policies.

Hosseinpour et al. [76] present a new approach to tackling big data management in a fog computing context. It relies on the concept of smart data which is inspired by ADB. The intelligent information has a similar structure to the ADB. It is a smart unit which can evolve and participate in the operation of an IoT application. An essential and lightweight version of intelligent data is generated by IoT sensors. The smart data construct develops (grows) when it travels through the network towards the cloud, merging with other cells. The process is the opposite when data moves from the cloud towards the actuators. The smart data concept focuses on reducing computational and network overhead resulting from handling big data in IoT rather than protecting these data.

Our earlier work [17] proposes a solution for protecting sensitive data in IoT by enforcing data owners' privacy policies based on ADB. The ADB scheme empowers data owners to enforce their privacy policies on their sensitive data throughout the entire lifecycle of the data. This explicitly addresses the issue of the lack of control over data that leaves its source. With the ADB scheme, "raw" or "passive" data are not available for anything or anybody; instead, data can be accessed only after the policy-enforcing VM "filters" them through all privacy policies. There are two main goals of using ADBs in the IoT: protecting subjects' sensitive IoT data from unauthorized disclosures and limiting access to subjects' sensitive data by authorized parties only to data satisfying the need-to-know principle. We outline the ADB lifecycle in IoT by describing sequential ADB stages including ADB creation, ADB dissemination, and ADB enabling. Once the

ADB reaches a destination host, the VM utilizes the host's hardware and operating system to run the ADB enforcement engine that enforces privacy policies for the ADB's data.

The ADB is a highly secure and privacy-preserving approach. However, it is computationally demanding due to the need for extra processing time during creating and enabling, as well as the extra size added to the data for policies and VM which increases the time for ADB transmission. This drives most of the researchers, in the literature, to use ADB for environments that do not require real-time data processing. In our solution, we aim to take advantage of the high level of security and privacy provided by ADB for IoT, on the side. On the other side, overcoming the abovementioned ADB issue. For that, we integrate ADB with fog computing. The novelty of our solution falls in using the promising fog computing paradigm, which is mainly proposed for real-time IoT processing, to handle the processing of ADB. This clearly distinguishes our solution from others. Fog nodes designed to work in a collaborative and distribution manner by using dedicated resource management policies [77]. These policies assure that processing assigned to a fog node within its capability to perform this processing in real-time and the remaining processing assigned to alternative fog node and so on. This assures that ADB can be handled by fog nodes without breaking the real-time constraints. We also utilize data minimization feature to assure that whenever ADB executed by fog nodes, its size decreases due to data minimization by the mean of partial or null data disclosure.

CHAPTER 3

THE PROPOSED SOLUTION

3.1. Introduction

In this chapter, we introduce our proposed solution for protecting the privacy of sensitive IoT data. By proposing a solution, we address the *problem statement* and prove the *research hypothesis*, which are stated in Subsections 1.3 and 1.4, respectively. This chapter presents an overview of the proposed policy enforcement fog module (PEFM) solution in terms of its major functionalities for handling sensitive IoT data and its major components that perform these functionalities. The chapter then discusses in detail the system design for the PEFM starting by its high-level design followed by the system design for its two major submodules LPEM and RPEM. For each submodule, the chapter describes the structural components and the process flow that shows the interactions between the module components to achieve its functionality for protecting sensitive IoT data.

3.2. Overview of PEFM

The proposed solution, called Policy Enforcement Fog Module (PEFM), is a software module for protecting sensitive IoT data throughout their entire lifecycle using the power of policy enforcement in the fog. The PEFM should be placed within local fog nodes in IoT, as closely as possible to the sensors that generate data [78]. The placement of policy enforcement in a local fog

enables PEFM to act on data in their local domain where they are generated by IoT sensors. Local domains can usually be considered by data owners as trusted [79]. PEFM can also act on data while they are in remote domains (remote fog nodes, clouds, hosts, etc.), which are usually considered by data owners as untrusted domains.

3.2.1. Real-Time and Non-Real-Time Processing of IoT Data

The Real-Time Constraints (RTC) defined as a set of critical timing requirements that must be met by the system for real-time applications [80]. Using or not using RTC distinguishes between real-time processing and non-real-time processing. We assume that real-time processing of IoT data obtained at the edge can be done mostly within the local fog (the portion of the fog closest to the edge). By this assumption, any processing of IoT data within the remote fog, the cloud or other nodes might violate the RTC. Hence, we distinguish two forms of data processing in IoT from the fog computing point of view: (1) processing data completely at the edge, which is required for most of the real-time applications that are run locally without using remote servers or Cloud and is allowed for non-real-time applications that are able to run locally; and (2) preprocessing data and disseminating them to a remote server or Cloud for more processing and long-term storage, which is applicable to non-real-time applications that are run remotely.

3.2.2. Local and Remote Policy Enforcement

PEFM performs local and remote policy enforcement using Local Policy Enforcement Module (LPEM) and Remote Policy Enforcement Module (RPEM), respectively.

LPEM performs direct privacy policy enforcement for sensitive data provided to local IoT applications (including real-time data and real-time applications, resp.). In this way, sensitive data

fed from LPEM to local IoT applications are guaranteed to satisfy privacy policies defined for these data. Depending on the policies, some of these data are never shown to some local applications, while others might be shown to some local applications after privacy-preserving transformations (such as data masking or de-identification).

RPEM sets up a mechanism for indirectly enforcing privacy policies for sensitive data disseminated to remote IoT applications (by our assumption, these can be only non-real-time data and non-real-time applications, respectively). This mechanism is based on using Active Data Bundles (ADBs), which include privacy policies for the data they carry (as described in Introduction). When any remote IoT application (or other IoT entity, such as a host or a user) tries to access data carried by an ADB, the ADB's VM activates the process of privacy policy enforcement. As was the case for LPEM, depending on the policies, some disseminated data are never shown to some IoT entities, while others might be shown to some IoT entities after privacy-preserving transformations (such as data masking or de-identification).

PEFM is a system module, that can be integrated with an existing IoT system to protect sensitive data generated by IoT sensors. PEFM can work either in a centralized manner (when there is no need for collaboration with other fog nodes) or a distributed manner (otherwise).

3.3. System Design

This section provides a high-level conceptual design for the PEFM solution—including design for its major components, namely, LPEM and RPEM. The PEFM design, illustrated in the system environment context in Figure 3.1, shows PEFM components, their deployment in the EFN middleware, and their interactions with IoT sensors (data sources), local and remote IoT applications (data destinations). PEFM includes the following components:

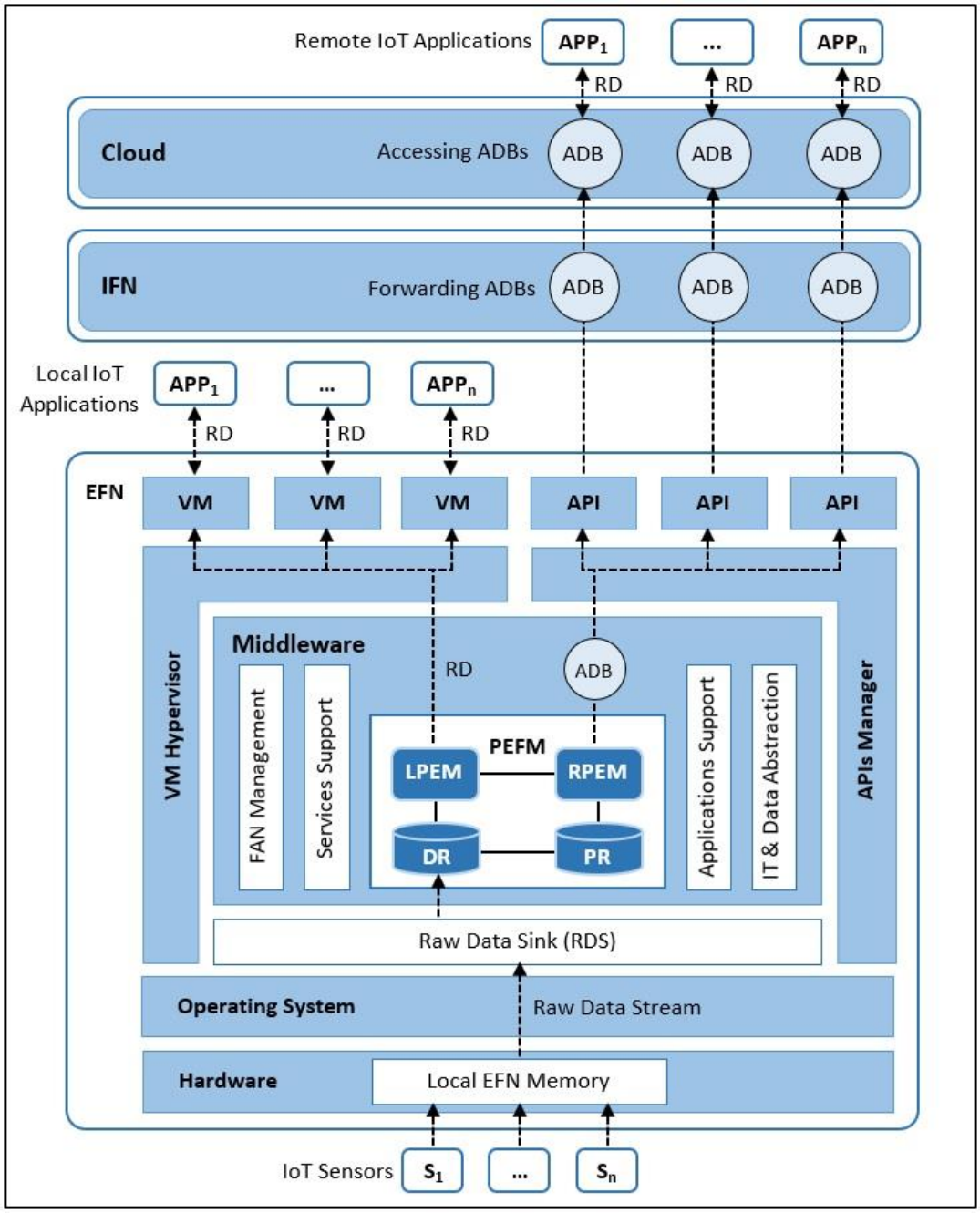

Figure 3.1. The high-level structure of PEFM within EFN and its IoT system.

1) *Data Repository (DR)*—stores data collected from IoT devices; these data start their lifecycle in their owner's domain. DR stores data for a single fog node; these are either data collected locally by this fog node, or data from remote nodes.

2) *Policy Repository (PR)*—stores all privacy policies for a single fog node, which might be defined by a data owner (using templates, and other assistance) or by a data guardian (e.g., system administrator) with the consent of a data owner.

3) *Local Policy Enforcement Module (LPEM)*—enforces privacy policies directly for sensitive IoT data accessed by local applications within local fog domain.

4) *Remote Policy Enforcement Module (RPEM)*—enforces privacy policies indirectly for sensitive IoT data accessed by remote applications within remote fog or cloud domains.

The PEFM components are to be deployed and integrated into the middleware of an EFN. The IoT sensors, connected to an EFN, generate sensing data and send them to this EFN. EFN then stores these data in its local memory. While all data (incl. public and sensitive, a.k.a. private) received by EFN, only sensitive data delivered to PEFM (recall our research scope, Section 1.5) and stored automatically in the PEFM's DR. Privacy policies defined for sensitive data stored in DR are available in PR and ready for LPEM and RPEM.

EFN runs PEFM's LPEM for data generated for local IoT applications (including real-time IoT applications), which in turn enforces privacy policies directly for these data. LPEM then passes released data (RD) (data after policy enforcement) to users via their local IoT applications. (LPEM is described in more detail later.)

EFN runs PEFM's RPEM for data generated for remote IoT applications (including some non-real-time IoT applications), which in turn indirectly enforces privacy policies for these data by creating ADBs (carrying data and their privacy policies) and disseminating ADBs to remote nodes. ADBs (created by RPEM) reach the cloud and perform privacy policy enforcement (as a part of ADB enabling). Then, each ADB passes its RD to users via their remote IoT applications. (RPEM is described in more detail later).

Figure 3.2 shows a high-level flow diagram for the PEFM process. (Processes for LPEM and RPEM will be discussed in the two following subsections.)

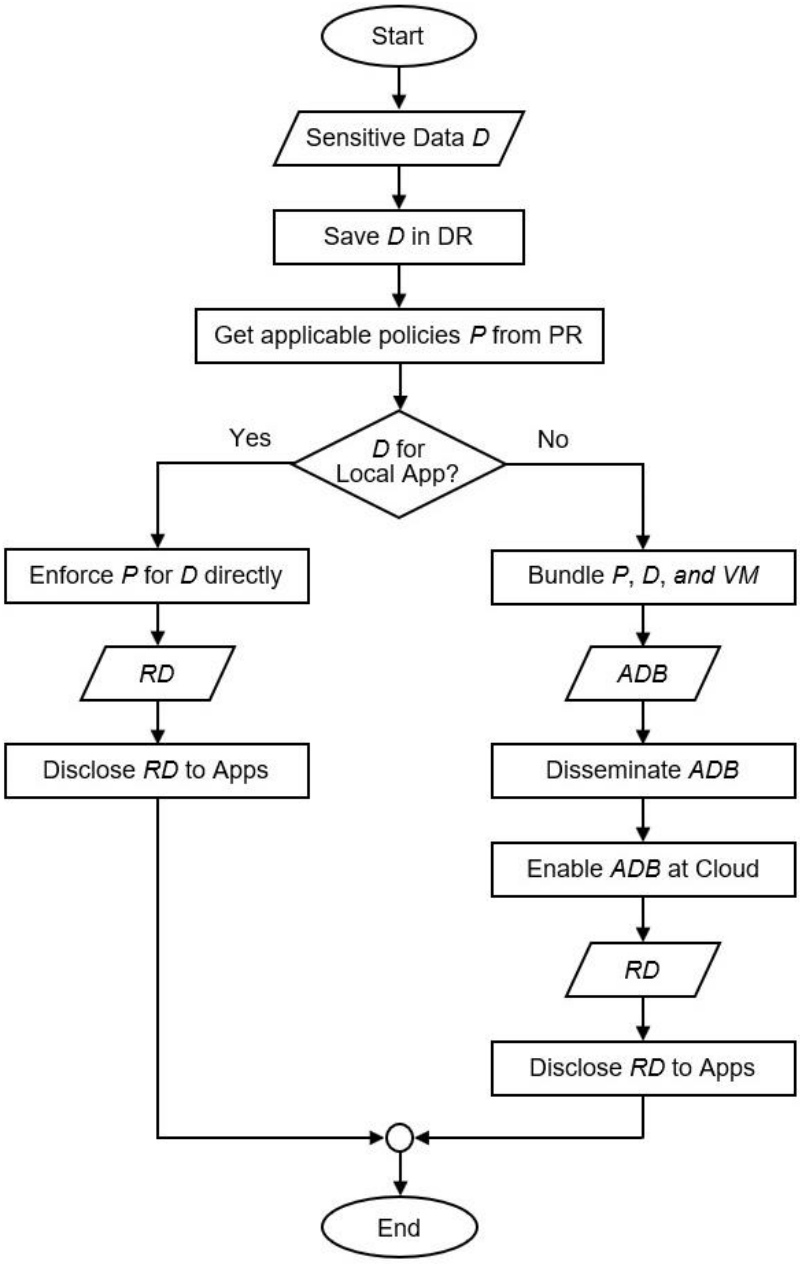

Figure 3.2. A high-level flow diagram for PEFM process.

For a given set of IoT sensors *S*, a fog node locally collecting data from *S* is an EFN for *S*. Any other fog node than this EFN is an IFN for *S*. This means EFN for *S1* is an IFN for another set *S2* of IoT sensors. Similarly, recall our discussion of the two kinds of IoT applications from the point of view of a given fog node *N*. An application running within *N* is its *local application*, and any application running on any node other than *N* is a *remote application* for *N*. This means that for a given EFN, any application running on IFN or Cloud is a remote application for EFN and data should be sent to this application encapsulated by an ADB [81]. Therefore, ADB created by EFN, forwarded by IFN (shown in Figure 3.1) or enabled by IFN (not shown in Figure 3.1) or enabled by Cloud.

3.3.1. LPEM System Design

Local Policy Enforcement Module (LPEM) is developed for IoT on the basis of the eXtensible Access Control Markup Language (XACML) reference architecture [82], [83]. In LPEM, we adopt an approach to enforcing privacy policies, in which data owners can specify and enforce locally (via LPEM within the local fog node) privacy policies for their sensitive IoT data. The PEFM's LPEM consists of software system subcomponents that are work in a collaborative manner to perform the policy enforcement process for raw IoT data. The major subcomponents of the LPEM module, inherited from the XACML architecture, shown in Figure 3.3, are:

1) *Policy Resolving Point (PRP)*—gets sensitive data items from DR, and obtains the applicable privacy policies from PR.

2) *Policy Information Point (PIP)*—provides information on purpose, visibility, granularity, and obligations for data usage.

3) *Policy Decision Point (PDP)*—makes data access decisions based on the applicable privacy policies from PRP and policy information from PIP.

4) *Policy Enforcement Point (PEP)*—enforces the data access decisions made by PDP by passing only released data RD to local applications.

Figure 3.3 shows PRP, PIP, PDP, and PEP as an integral part constituting the LPEM module that is in turn integrated within EFN's Middleware. System administrator (data guardian) acts on specifying data owner's privacy policies for specific data types for specific IoT application. For example, privacy policies for surveillance data different than those for energy data. Data user who requests the data or whom the data collected for. Figure 3.4 outlines the LPEM's algorithm for the local policy enforcement process.

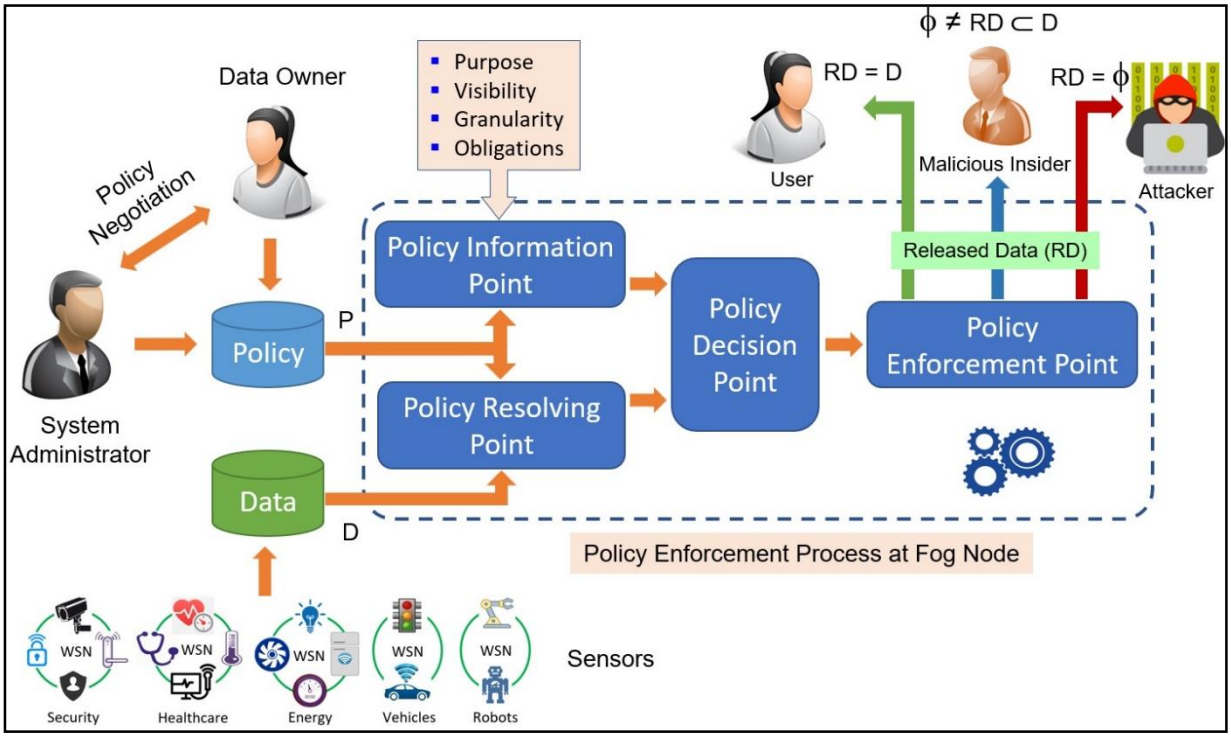

Figure 3.3. LPEM Structure and process flow.

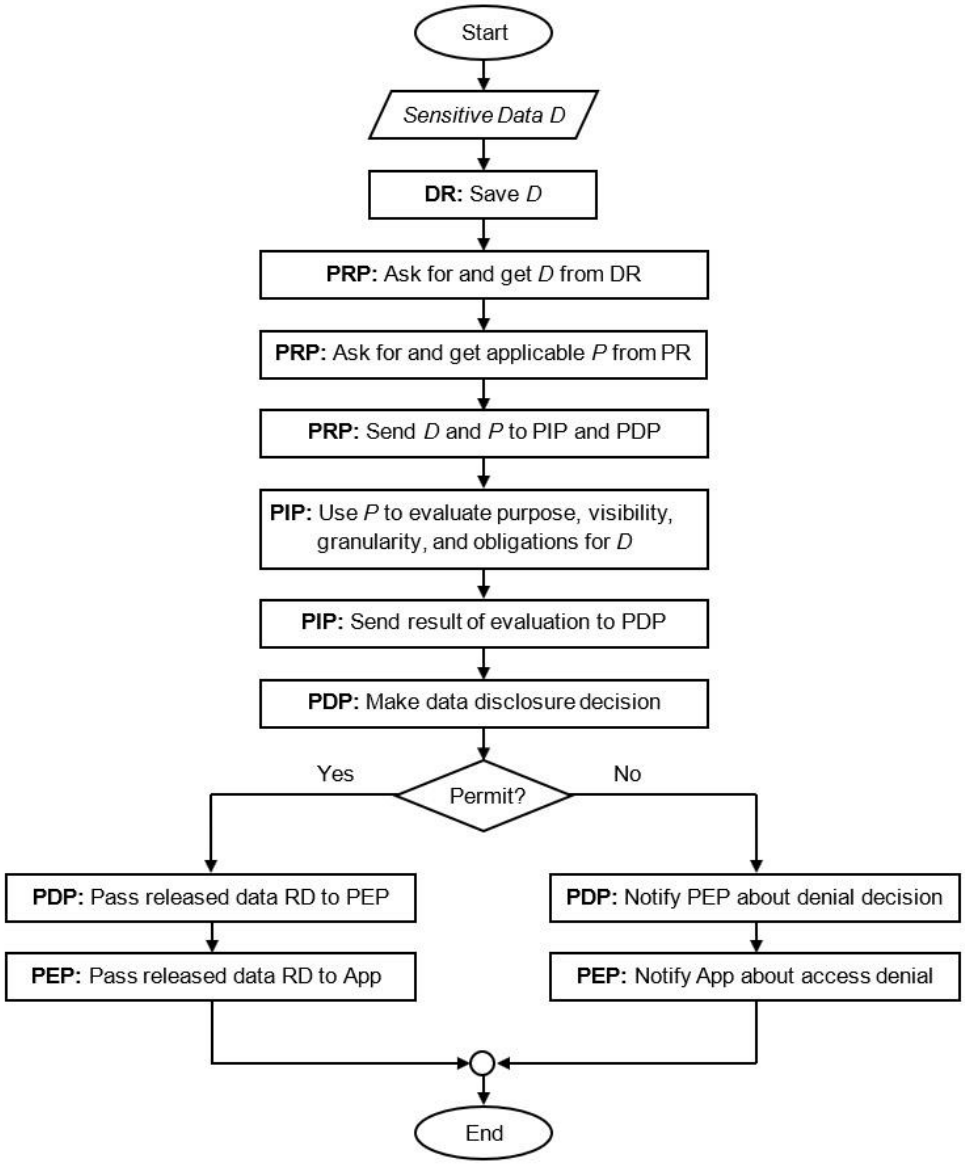

Figure 3.4. Flow diagram for LPEM process.

LPEM integrated into the EFN's Middleware is a mediator for passing sensitive IoT data among data sources and data destinations. LPEM assures that no data is delivered to users without enforcing data owner's policies on these data. LPEM is activated when new data comes to DR from a Raw Data Sink (RDS) running on EFN for processing a raw data stream from IoT Sensors. The LPEM's local policy enforcement process includes the following sequence of steps: (1)

sensitive data D received by DR from IoT sensors via a specific RDS; (2) PRP requests for D from DR; (3) PRP gets D from DR; (4) PRP requests for all privacy policies P applicable to data D; (5) PRP gets P from PR; (6) PRP sends D and P to PDP and PIP; (7) PIP evaluates the purpose, visibility, granularity, and obligations for specific D and P and sends the result of the evaluation to PDP; (8) PDP uses privacy policies to decide whether D should be fully disclosed, partially disclosed, or denied (not disclosed). Then, PDP sends to PEP data allowed by its decision; these are released data (RD); and (9) PEP passes RD to users via local IoT applications.

3.3.2. RPEM System Design

On the basis of the Active Data Bundle (ADB) scheme [24] and the XACML reference architecture, we developed a Remote Policy Enforcement Module (LPEM) for IoT. In RPEM, we adopt a self-protecting data approach using the power of ADBs, in which data owners can specify their privacy policies for their sensitive IoT data locally within local fog nodes and enforce these policies remotely within remote fogs or Cloud. The major subcomponents in the RPEM module, as shown in Figure 3.5, are:

(1) *Policy Resolving Point (PRP)*—gets the sensitive data items from DR and obtains the applicable privacy policies from PR.

(2) *Policy Information Point (PIP)*—provides information on purpose, visibility, granularity, and obligations for data usage.

(3) *ADB Creation and Dissemination Point (ADB-CDP)*— construct a complete ADB including PDP and PEP (inherited from XACML) in ADB's VM and disseminates this ADB to remote nodes.

Figure 3.5 also shows the process flow for the remote policy enforcement process done by RPEM within EFN, IFN, and Cloud. RPEM integrated into the EFN's Middleware is a mediator for setting up a mechanism for protecting sensitive IoT data whenever they accessed by remote nodes. RPEM assures that no data is delivered to a remote user without enforcing data owner's policies on these data. Figure 3.6 outlines the RPEM's algorithm for the remote policy enforcement.

RPEM is activated when new data comes to DR from an RDS running on EFN for processing a raw data stream from IoT sensors. The RPEM's remote policy enforcement process involves the following sequence of steps: (1) sensitive data D received from IoT sensors via a specific RDS; (2) PRP requests for D from DR; (3) PRP gets D from DR; (4) PRP requests for all privacy policies P applicable to data D;

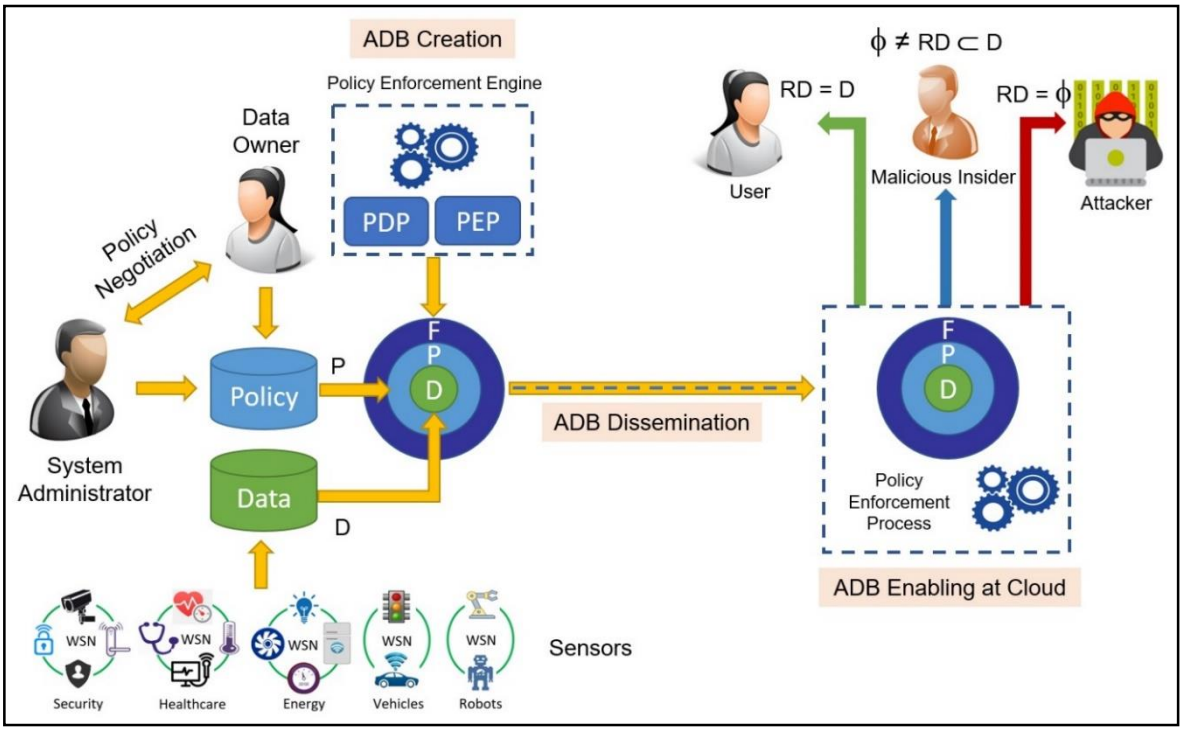

Figure 3.5. RPEM structure and process flow.

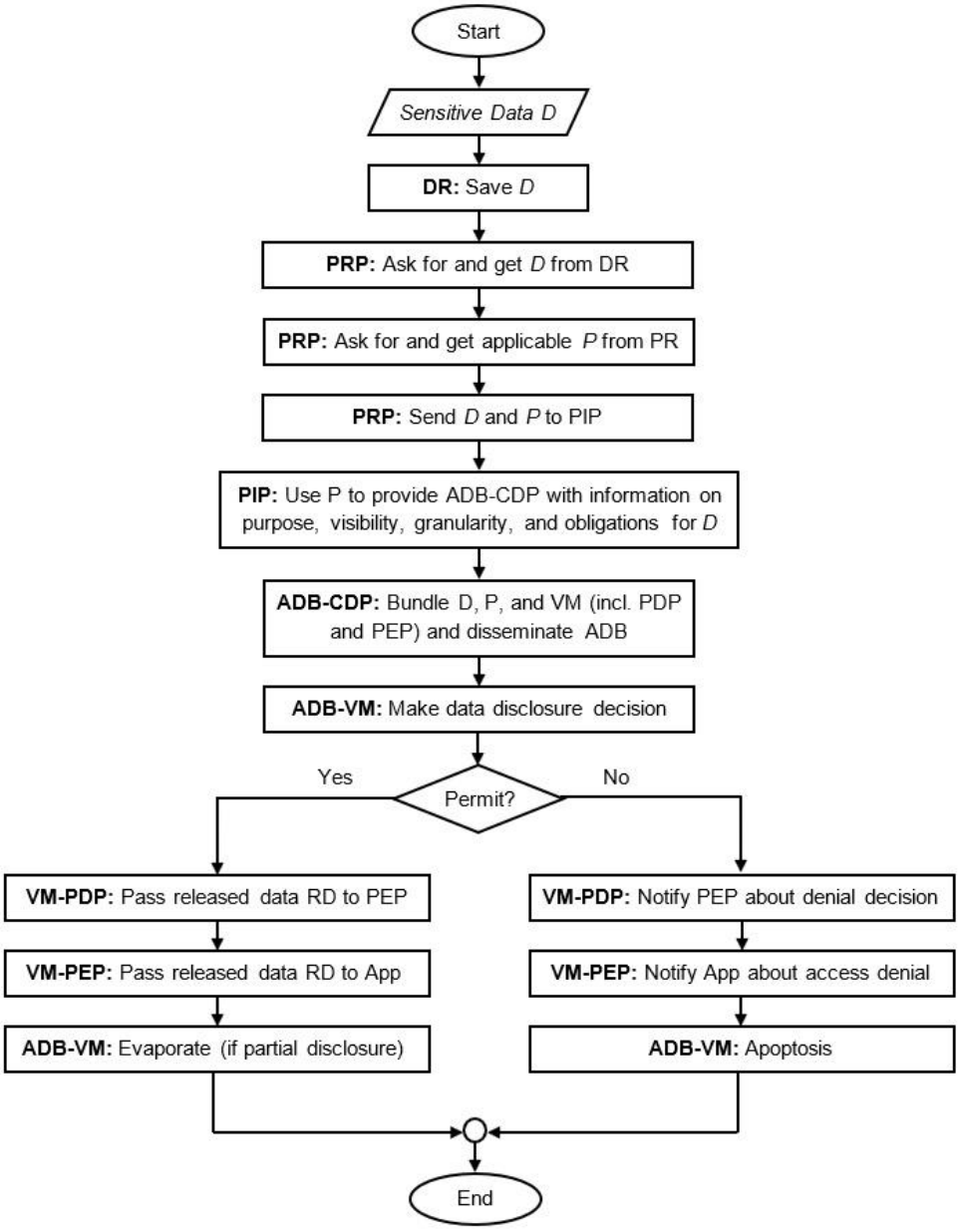

Figure 3.6. Flow diagram for RPEM process.

(5) PRP gets P from PR; (6) PRP sends D and P to PIP and ADB-CDP; (7) PIP uses P to provide ADB-CDP with information on purpose, visibility, granularity, and obligations for specific D; (8) ADB-CDP first constructs a complete ADB that encapsulates sensitive data with metadata, including all privacy policies required to protect these data, and VM as an engine able to enforce

privacy policies whenever ADB accessed by any visited hosts. ADB's VM must include PDP and PEP functionality; (9) ADB-CDP sends ADB to a destination host; (10) at the destination host in the cloud, PDP in ADB's VM can decide whether D should be fully disclosed, partially disclosed, or not disclosed, that is, PDP can produce and send to PEP in ADB's VM the appropriate released data RD; and (11) PEP in ADB's VM's passes RD to the user via the remote application.

3.4. Chapter Conclusions

In this chapter, we have provided an overview of the proposed PEFM solution and have developed a high-level conceptual design for it illustrating its structural components, and their interactions with IoT entities. Then, for both major PEFM components, namely LPEM and RPEM, we have presented system design emphasizing their structure and the process flow that shows the interaction among their structural blocks and the IoT environment.

CHAPTER 4

PROOF OF CONCEPT CASE STUDY

4.1. Introduction

This chapter introduces our proof-of-concept case study Foscam Home Surveillance System (FHSS) [84]. The chapter first gives a description of the FHSS system including the technical specifications of its devices (to be used later for simulation design). Then, the chapter identifies privacy threats in the existing FHSS case study. The framework of using the proposed PEFM system for FHSS is then presented. PEFM is introduced in this chapter as a privacy control for FHSS data. Last, the chapter discusses eight different scenarios for the case study in terms of privacy risk levels for both local and remote FHSS data processing. The privacy risk is defined as the probability of there being existing attackers or maliciuos insiders in the FHSS system.

4.2. Description of Foscam Home Surveillance System

FHSS is an automation system able to monitor a home environment (indoor and outdoor) using intelligent hardware and software subsystems with limited human intervention or without any human intervention [85]. FHSS is a commericial home security system that is recognized as the best home surveillance system of 2018 [86]. It is integrated with Smart Home (SH) technology which is an IoT-enabled home equipped with sensing, actuating, monitoring, and controlling devices that can be controlled remotely by a smartphone, tablet, or computer. The major task of

FHSS is providing safety services for home residents as well as assuring their security through erroneous event detection. FHSS includes setting up a network of real-time, high-definition surveillance cameras in the home to be monitored either by third parties or by homeowners themselves by means of smartphone apps. There are many types of cameras used for this purpose such as indoor cameras, outdoor cameras, mobile cameras, and covert or hidden cameras. These cameras can provide smart notification, night vision, etc. FHSS enables a wide variety of applications and services for SH, including home security, home safety, home automation, baby monitoring, pet watching, eldercare, and remote viewing and control. The following subsections present the description of the major FHSS devices.

4.2.1. Foscam IP Camera

The major data source in the FHSS is the IP Camera (IPC) which monitors a given surveillance area within its fields of view [87]. Multiple IPCs are distributed over multiple surveillance areas within a Smart Home (SH). The Foscam E1 IPC [88] is a 1/2.9" CMOS sensor type that generates H.264 video format with 1080P (1920*1080) & 720P (1280*720) quality. It provides 25 ft IR range with 110° view angel. The E1 IPC uses IEEE 802.11b/g/n wireless standard with data rates 11 Mbps, 54 Mbps, and 150 Mbps for IEEE 802.11b, IEEE 802.11g, and IEEE 802.11n, respectively.

4.2.2. Foscam Base Station

An IPC sends a Raw Video Stream (RVS) to the SH base station (BS) which is a central hub connecting the IPC to the Internet. The BS acts as an Edge Fog Node (EFN) in our case study that is a part of the local fog and connecting the SH's IPCs with the cloud. The Foscam E1 BS [88]

has 720P &1080P recording resolution and H.264 compression format. It provides a single channel 720P & 1080P synchronous video playback. The E1 BS also uses IEEE 802.11b/g/n wireless standard with 1× 10/100 Mbps RJ45 port network interface. It supports IOS, Android, and any 3G/4G Smartphones.

4.3. Privacy Threats in FHSS

The surveillance cameras watching humans (e.g., in security, childcare, eldercare, etc. applications) can be used in combination with other connected devices. Together they collect images and other data that may be captured in privacy attacks on the monitored people. In this way, FHSS increase the risks of a purpose drift: information collected for one purpose is used for another purpose [89]. Christin et al. [90] investigated privacy risks associated with sharing environmental surveillance data. They mention that the contents of the captured images are most likely revealing personal (sensitive) information about the people in that environment.

4.4. Framework of Using PEFM for FHSS

To address privacy threats in the FHSS, we integrate the proposed PEFM system in the FHSS as a privacy control. Figure 4.1 shows the framework of using PEFM for FHSS. The BS receives RVS packets from IPCs connected to it. Only sensitive RVSs are saved on the PEFM's DR to be ready for policy enforcement. In the case of sending data to the user within a local home domain, the PEFM's LPEM sends local notifications after performing local policy enforcement on them as Local Released Data (LRD) to the user.

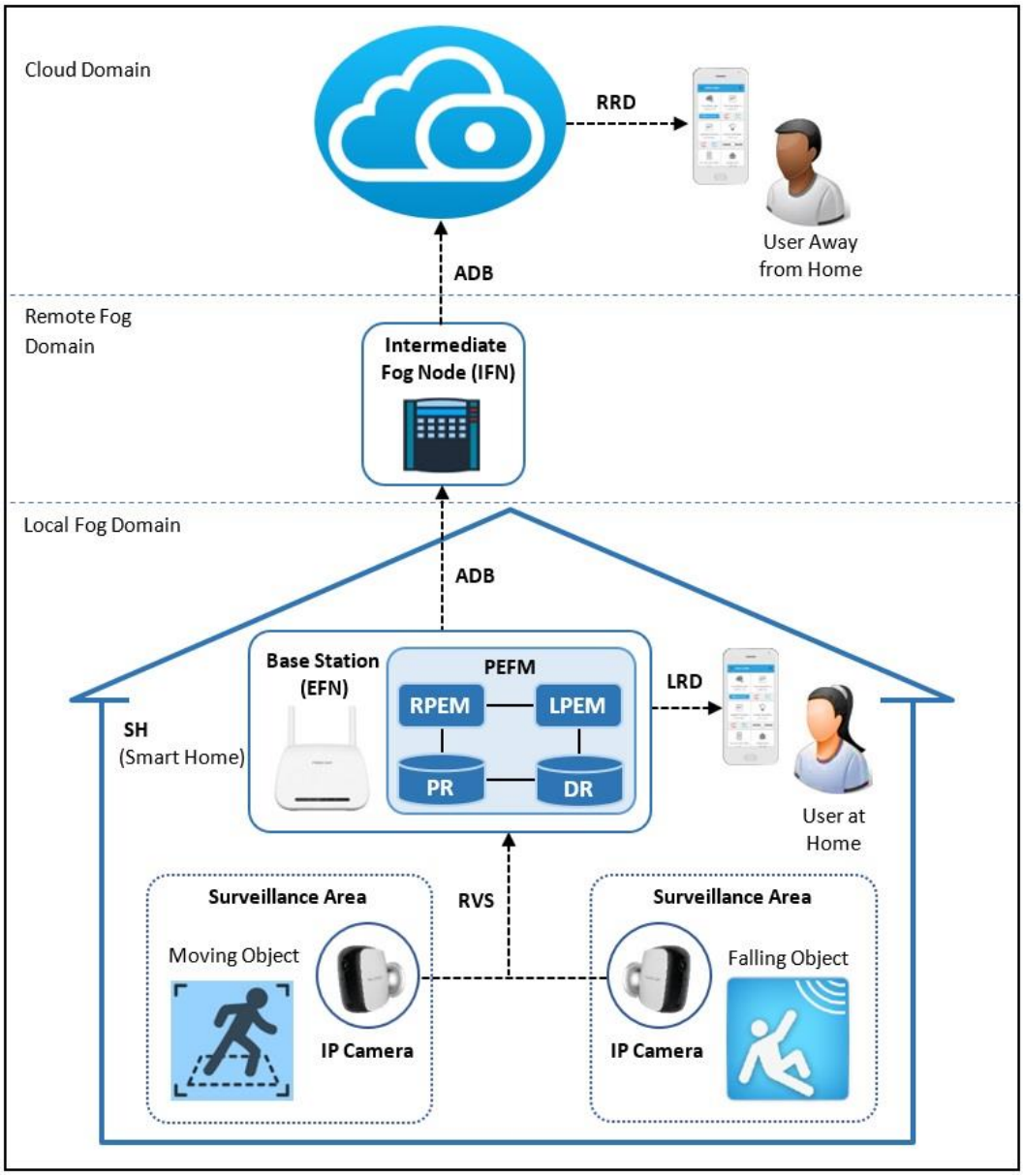

Figure 4.1. Framework of using PEFM for FHSS.

Local notifications include processed video streams that show the moving objects within a certain surveillance area to the user as part of the notification. For example, LUD enables home resident to identify the person who is in her home area by showing this person in the content of the video within the received LRD. In our scenario, LUD does not perform processing on the received LRD, it just displays notification data to the end users. In the case of sending data to the

user within a remote cloud domain, the PEFM's RPEM sends remote notifications in the form of ADBs which are disseminated to the cloud (among intermediate fog nodes) and then released to the user as Remote Released Data (RRD). Not that each RRD is already targeted for a user U who requested data. This is because all policies for RRD and U were enforced in the cloud using ADB-VM. RRD targeted for U need to be delivered to U in a secure way. For example, a strong encryption. We do not need to use ADB for the last hop on the rout to U since policies for data are already enforced and hence RRD are no more sensitive.

4.5. Scenarios of the FHSS-based PEFM Framework

To test the feasibility and efficiency of the PEFM system for assuring privacy in the FHSS case study, we consider different scenarios in terms of privacy risk of having (or note) malicious insiders, attackers, or both in the system. We categorize our scenarios into two major groups: Scenarios for local Foscam system—where LPEM is the privacy actor for them; and Scenarios for remote Foscam system—where RPEM is the privacy actor for them.

4.5.1. Local Foscam Scenarios

We consider four different scenarios for the local Foscam system, as shown in Figure 4.1. The PEFM's LPEM is the main policy enforcement actor for these scenarios.

Figure 4.2 shows *Home No-Risk Scenario (HNRS)* where the home residents Bob and Michelle are at home and they can access surveillance data locally. This scenario is free from attackers or malicious insiders, so it is considered as a *baseline* for local FHSS. In this scenario, LPEM enforces privacy policies even for normal users to maintain data access authorization by home residents who have different levels of authorities in accessing the FHSS data.

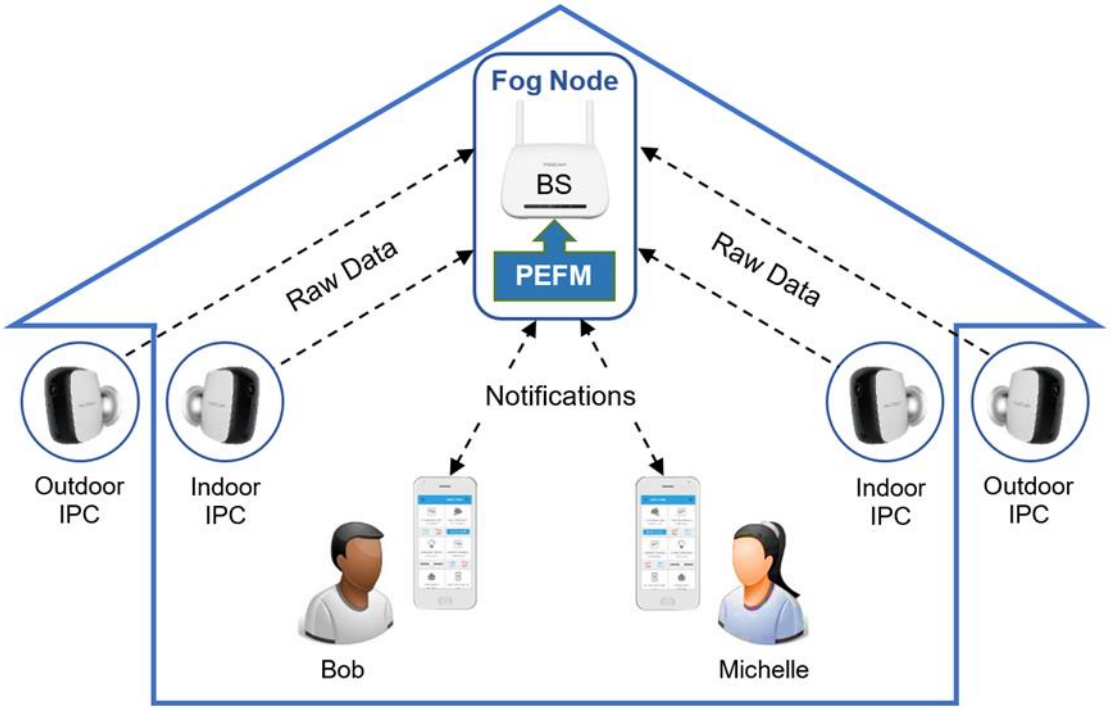

Figure 4.2. Home no-risk scenario.

Figure 4.3 shows *Home Malicious Insider Scenario (HMIS)* where Bob and Michelle have a technical issue in their FHSS, so they have called the Foscam customer service which in turn sends a technician to fix the issue. Now, the technician is considered in this scenario to be a data user which may or may not be a malicious insider. In both cases, access to home data should be controlled by PEFM's LPEM. Figure 4.4 shows *Home Attacker Scenario (HATS)* where Bob and Michelle are at home and there is an attacker who attempts to access the home surveillance data. PEFM's LPEM assures that this kind of access is denied by enforcing privacy policies on the attacker's access. Figure 4.5 shows *Home High-Risk Scenario (HHRS)* where Bob and Michelle have a technical issue in their FHSS, technician, which may or may not a malicious insider, comes fix the issue. At the same time there is an attacker who attempts to access the home surveillance data. PEFM's LPEM assures that access by the technician is controlled and access by attacker is denied by enforcing privacy policies.

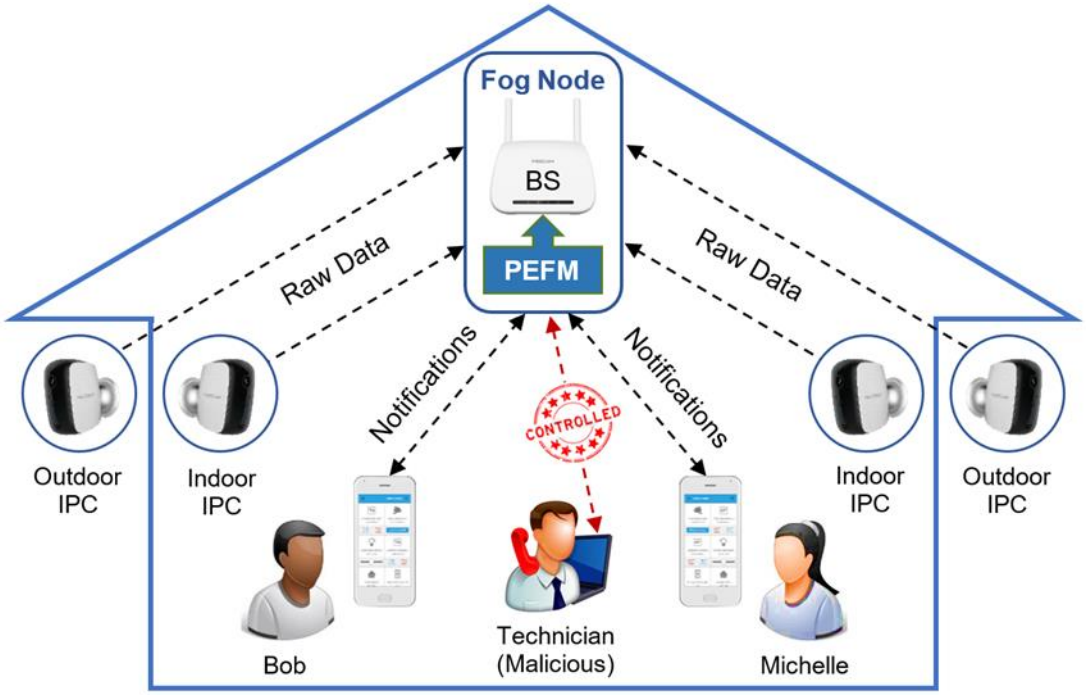

Figure 4.3. Home malicious insider scenario.

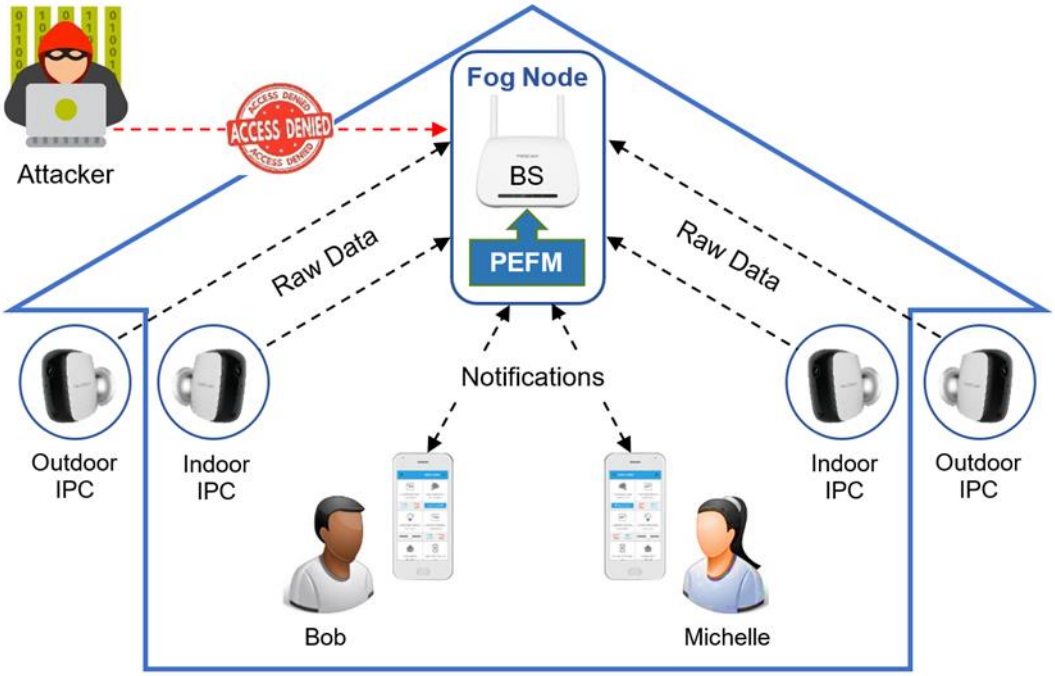

Figure 4.4. Home attacker scenario.

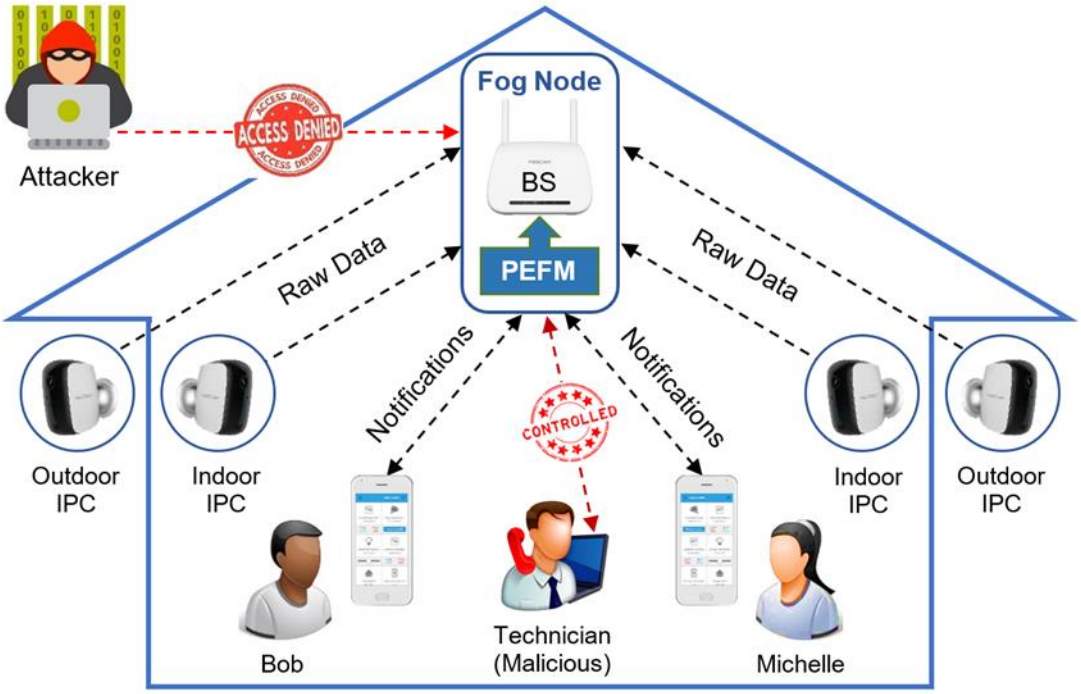

Figure 4.5. Home high-risk scenario.

4.5.2. Remote Foscam Scenarios

We also consider four different scenarios for the remote Foscam system. The PEFM's RPEM is the main policy enforcement actor for these scenarios.

Figure 4.6 shows *Cloud No-Risk Scenario (CNRS)* where Bob and Michelle are away from home, and they can access surveillance data remotely via the cloud. This scenario is free from attackers or malicious insiders, so it is considered as a baseline scenario for the remote system. Figure 4.7 shows *Cloud Malicious Insider Scenario (CMIS)* where a Foscam employee access home data remotely to fix an issue in the FHSS. This kind of access should be controlled by RPEM. Figure 4.8 shows *Cloud Attacker Scenario (CATS)* where an attacker attempts to access the home surveillance data via the cloud. This kind of access should be denied by RPEM via enforcing privacy policies on the attacker's access using the ADB.

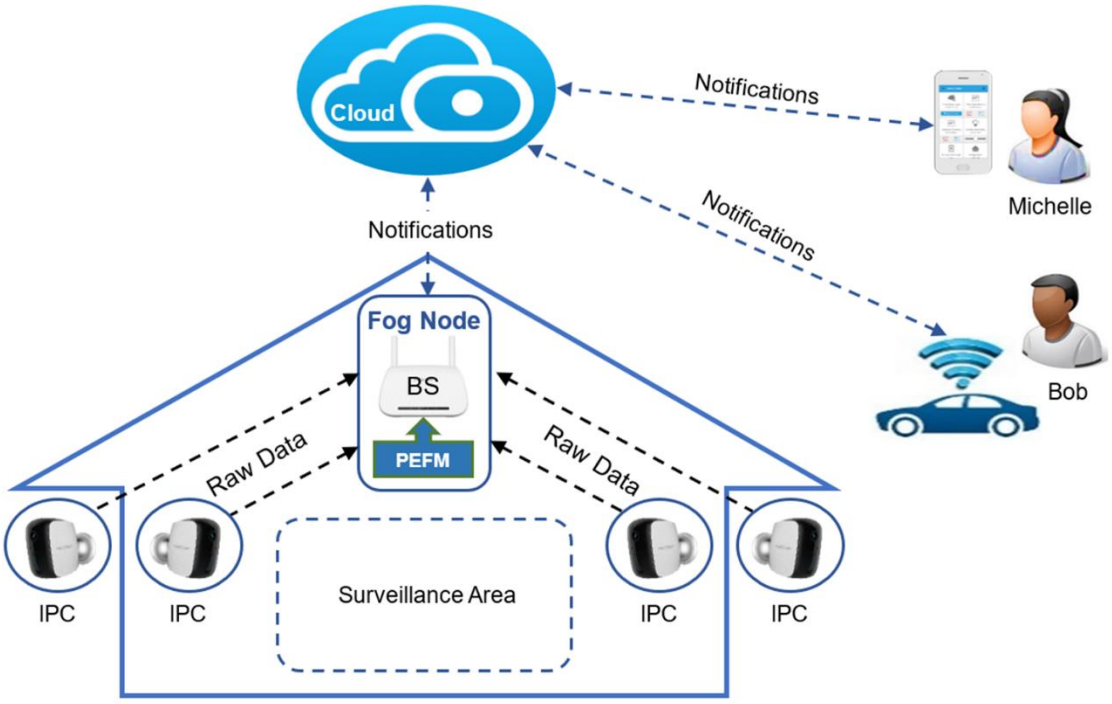

Figure 4.6. Cloud no-risk scenario.

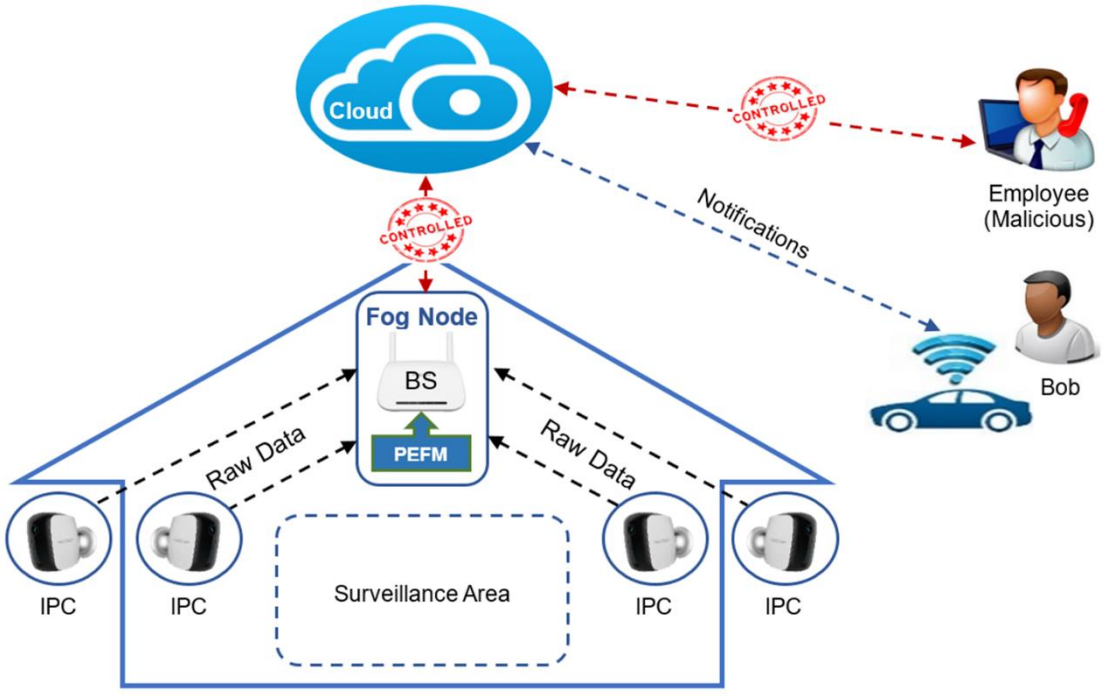

Figure 4.7. Cloud malicious insider scenario.

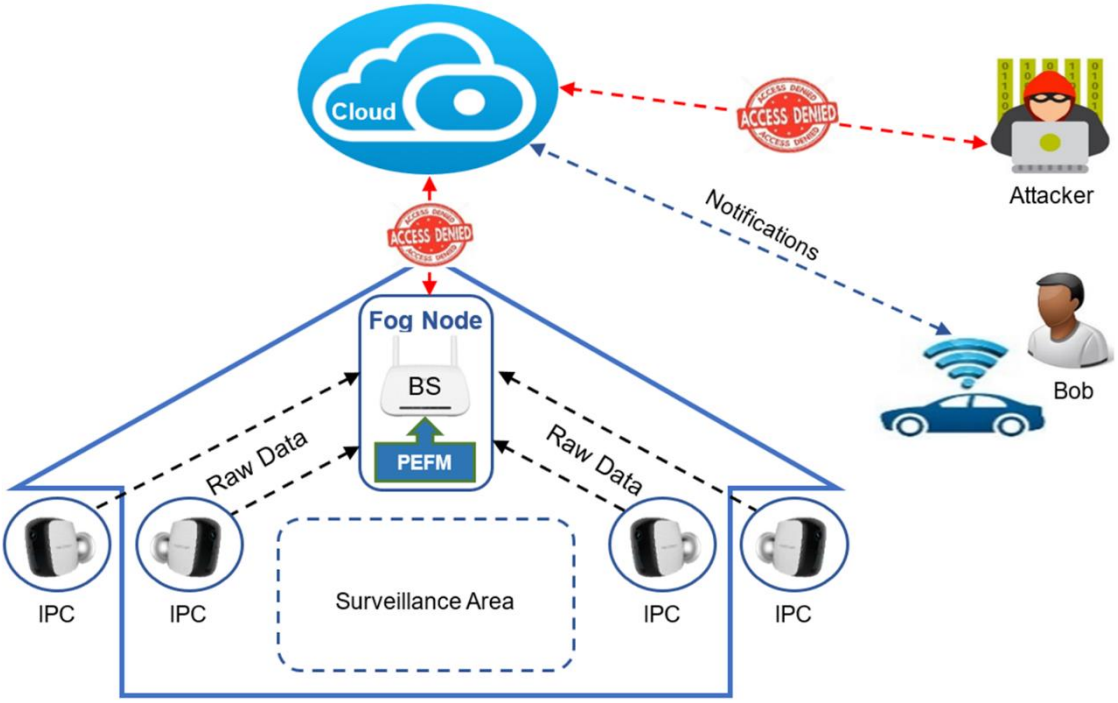

Figure 4.8. Cloud attacker scenario.

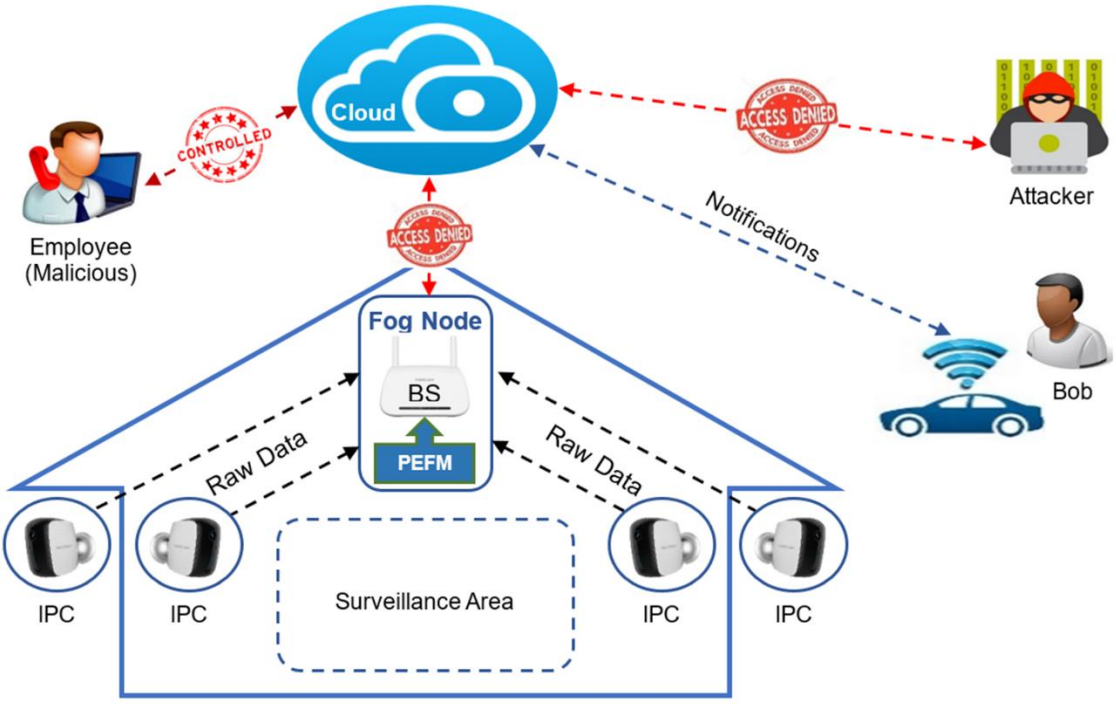

Figure 4.9. Cloud high-risk scenario.

Figure 4.9, above, shows *Cloud High-Risk Scenario (CHRS)* where Bob and Michelle have a technical issue in their FHSS, technician access FHSS data remotely via the cloud to fix the issue. Now, the technician is considered in this scenario to be a data user which may or may not be a malicious insider. At the same time, there is an attacker who attempts to access the home surveillance data remotely via the cloud too. RPEM assures that access by the technician is controlled and access by the attacker is denied by enforcing privacy policies.

4.6. Chapter Conclusion

In this chapter, we have introduced a proof-of-concept case study Foscam Home Surveillance System (FHSS) and have described its structure and its devices' technical specifications. Then, we have identified privacy threats in the existing FHSS case study and presented the framework for using the PEFM system as a privacy control for FHSS to address the identified threats. Lastly, we have discussed eight scenarios for the FHSS case study in terms of privacy risk levels for both local and remote FHSS data processing.

CHAPTER 5

SIMULATION DESIGN

5.1. Introduction

In this chapter, we develop a simulation to evaluate the framework of using policy enforcement fog module (PEFM) for the Foscam home surveillance system (FHSS)—our case study that is discussed in the previous chapter. Our discrete event simulation design includes four major parts. The first part includes stating the assumptions for the simulation and describing each assumption. The second part includes declaring the simulation parameters and their values as well as our justification for choosing certain values. The third part includes declaring the random simulation variables and their probability distributions, and the values for the distributions parameters, as well as our justification for choosing a certain distribution and certain values. The fourth and last part includes identifying the evaluation measures of the simulation and describing them and how they are calculated during simulation implementation as well as how the statistics for these measures are collected from the simulation outputs.

5.2. Simulation Assumptions

The assumptions for the simulation can be stated as follows:

A1) *Edge Fog Node (EFN) is the only IP Camera gateway*: EFN serves all incoming and outgoing data for the IPCs. This means that we do not consider the cases when IPCs

communicate their data directly with IFN or Cloud (without going through the smart home (SH's EFN)).

A2) *Single-packet messages*: Each message is sent as a single packet. In other words, we consider packet switching in our simulation networks instead of message switching where each message includes multiple packets [91].

A3) *Active Data Bundle sent as a packet:* An ADB sent in the simulation as a packet with a payload including sensitive data, metadata (including privacy policies), and an executable code as a policy enforcement engine [24].

A4) *All generated data are sensitive*: Raw surveillance data, generated by IPCs, as well as data sent by EFN to other nodes in the system are all sensitive. However, these data have different sensitivity levels (as we will describe later).

A5) *Processing nodes*: EFN, intermediate fog node (IFN), and Cloud are the only processing nodes. IPCs perform sensing and actuating only. User's smartphones are just receiving local released data (LRD) and remote released data (RRD) from EFN and Cloud, respectively.

A6) *Reliable links*: The EFN-to-local user device (LUD) and the Cloud-to-remote user device (RUD) links are so reliable that no messages acknowledgments from LUD to EFN and from RUD to Cloud are needed.

A7) *XACML privacy policy evaluation:* In general, evaluating XACML privacy policies produces four possible results: *Permit*, *Deny*, *Indeterminate*, or *Not Applicable* [83]. In our simulation, we consider only Permit (*partial* or *full* disclosure) and Deny (*null* disclosure).

5.3. Simulation Parameters

We consider constant parameters for network nodes, network links, and network configurations. According to that, this section classifies simulation parameters into three major categories: parameters for network nodes, parameters for network links, and parameters for network configurations. The parameters values are chosen based on the values commonly used in the literature [77], [92], [93]. However, some values are chosen somewhat arbitrarily but within a certain logic when no significant impact is envisioned.

5.3.1. Parameters for Network Nodes

In our simulation, we have three types of nodes: source nodes which are the IPCs in the FHSS system; processing nodes which are EFN, IFN, and Cloud; and destination nodes which are LUD and RUD. We define parameters for processing nodes only since the source nodes limited to sending data to the processing nodes, and the destination nodes are limited to receiving data from the processing nodes. For each processing node, we define CPU speed, which is the maximum number of instructions that a node's CPUs can process per second, defined in MIPS (Million Instructions Per Second) [94]; and RAM size, which is the size of the short-term memory of the network node defined in Gigabyte [95], as shown in Table 5.1. The values for CPU speed and RAM size parameter are based on the values used in the literature.

Table 5.1. Simulation parameters for network nodes.

Node	CPU Speed (MIPS)	RAM Size (GB)
EFN	4000	4
IFN	10000	8
Cloud	44000	40

5.3.2. Parameters for Network Links

The network link is defined by its source and destination nodes. For each link type in the simulation, we define the following parameters, as shown in Table 5.2:

1) *Bandwidth* is the maximum number of bits per second that can be transmitted via the link, defined by Mbps (Megabits per second). The bandwidth values for links with source nodes IPC and EFN are from the specifications of Foscam's IPC and BS E1 Model [88]. The values for other types of links are based on the values used in the literature.
2) *Propagation delay* is a period from the instant when the last bit of a packet is placed onto a transmission link by the sending node till the instant when the last bit of the packet is received by the receiving node. The values of this parameter are also based on the values used in the literature.
3) *Link Parallelism* is the number of packets that can be transmitted simultaneously via the link. Mirtchev [96] defines the term *Packet-Level Link Capacity*. Since this name is not adequately descriptive, we use the more descriptive term "link parallelism." We arbitrarily chose different link parallelism values trying to reflect relative bandwidth of the links.

Table 5.2. Simulation parameters for network links.

Source	Destination	Bandwidth (Mbps)	Propagation Delay (ms)	Parallelism
IPC	EFN	150	1	4
EFN	LUD	100	1	2
EFN	IFN	100	8	2
IFN	Cloud	300	100	10
Cloud	RUD	1200	10	4

5.3.3. Parameters for Network Configurations

We study the performance of the proposed system by simulating five different network configurations named *Config 1* through *Config 5*. For each configuration, we define the following parameters, as shown in Table 5.3:

1) *EFN fan-in* is the number of IPCs providing input to EFN; in other words, the number of cameras per SH. Foscam allows up to 4 E1 IPC for a single home account [84]. Hence, the value of EFN fan-in is constantly equal to 4 for all configurations.
2) *EFN fan-out* is the number of LUDs that are getting output from EFN; in other words, several devices (e.g., smartphones) in SH that can receive LRD simultaneously. The value of this parameter is strictly related to the number of people per household in a certain country (e.g., in the USA) which is approximately equal to 3 [97]. Hence, the value of EFN fan-out is constantly equal to 3 for all configurations.
3) *IFN fan-in* is the number of EFNs providing input to IFN; in other words, number of SHs per IFN in a configuration. The value of this parameter can be a scale of any numbers based on the target of the study. We choose to increase the number of SH linearly with a constant delta equal to 20.
4) *Cloud fan-in* is the number of IFNs providing input to Cloud. We consider that for every 5 EFNs there is a single IFN receiving data from them in a configuration. Hence, we have the linearly increased number of IFNs with constant delta equal to 5.
5) *Cloud fan-out* is the number of RUDs getting output from Cloud. The values of this parameter are calculated because of multiplication of EFN fan-out with IFN fan-in.

Table 5.3. Parameters for network configurations.

Configuration	EFN fan-in	EFN fan-out	IFN fan-in	Cloud fan-in	Cloud fan-out
1	4	3	4	2	24
2	4	3	8	4	96
3	4	3	12	6	216
4	4	3	16	8	384
5	4	3	20	10	600

5.4. Random Simulation Variables

This section classifies random variables, used in the simulation, into four major categories: variables for packets, variables for policies, variables for risk levels, and variables for data sensitivity levels. For each category, we list the random variables and outline the distribution used as well as the values for the distribution parameters. The distributions of the simulation variables discussed in this section are chosen based on the distributions used in the literature [98] – [101]. However, for each distribution choice, we give our justification for the appropriateness of the chosen distribution to the simulated FHSS system. The value ranges that represent the parameters for the distributions are based on the values used in the literature [77], [92], [93]. However, some values are chosen arbitrarily.

5.4.1. Random Variables for Packets

Packets are the major processing and communication units in the simulated FHSS system which has four types of packets are RVS, LRD, ADB, and RRD. IPC is the source of the raw data (RVS) while other types of packets are resulted from processing the source RVS packets.

We model IPC as a discrete event generation sensor, which generates RVS packets when an event (such as movement or falling) occurred in a certain surveillance area. This means IPCs are not continuously sending raw data to the system network as a normal (not intelligent) cameras. Hence, there are different RVS arrival times, so we model RVS arrivals in our simulation as a random variable with the Poisson distribution of Lambda equal to 10, this value is chosen arbitrarily. The Poisson distribution is particularly appropriate if the arrivals are from a number of independent sources in a certain time interval [98], [99], which is the case in our simulated system.

Based on the above discussion, in a certain time interval, each IPC generates a certain number of RVS packets. Hence, we model the number of RVS packets generated by each IPC as a random variable with discrete uniform distribution in the range value from 5 to 15; these values are chosen arbitrarily. The discrete uniform distribution is typically used when it is believed that the value is equally likely over a bounded interval [98]. The discrete uniform distribution function takes two input numbers *m* and *n*; m is a lower limit, and it must be an integer, *n* is an upper limit, and it must be an integer greater than *m* [100].

In a realistic FHSS, packets vary in terms of the payload (content data) they carry. Therefore, we model packet size (for each packet type) as a random variable with Poisson distribution, as shown in Table 5.4.

Consequently, if the packets sizes are random, the efforts required for processing these packets are also random. Therefore, we model packet processing efforts (for each packet type) as a random variable with Binomial distribution, as shown in Table 5.4. The Binomial distribution is typically used to model the number of successes in a sequence of *n* independent and identical Bernoulli trials [101]. In our simulation, we need to model several identical instructions that successfully handle the processing of a certain type of packet. Hence, we believe that binomial

distribution is the best fit for this case. The binomial distribution function takes two input numbers n and p; n is a positive integer represents the parameter of the distribution, and p is a float number represents the probability of the distribution. It returns a positive integer number. The values for packet processing efforts, as shown in Table 5.4, are based on the values in the literature.

Table 5.4. Random variables for Packets.

Packet Type	Packet Size		Processing Effort (MI)		
	Distribution	Value	Distribution	Value	
		λ		*n*	*p*
RVS	Poisson	20	Binomial	300	0.75
LRD	Poisson	10	Binomial	150	0.85
ADB	Poisson	15	Binomial	200	0.75
RRD	Poisson	10	Binomial	150	0.85

5.4.2. Random Variables for Selective Data Disclosure

Selective Disclosure (SD) for IoT data is one of our privacy goals to be achieved by the proposed PEFM system. SD can be done based on data sensitivity level. We consider five numerical levels for data sensitivity are starting from number one representing the lowest level and ending with number five representing the highest level. For each sensitivity level, there are three corresponding value ranges. Each range represents the probability of one of the major three data disclosure decisions: full, partial, and null disclosures. In Table 5.5, the first two columns from the right show the value ranges for the five sensitivity levels, for these values, we divide the scale from 0% to 100% equally for the five levels. Note that these values are used to identify each level, and they are not random variables in the simulation. Table 5.5 also shows the random variables for the values of data disclosure types for each sensitivity level. These value ranges are

chosen arbitrarily by us. However, they reflect one major logic that is when sensitivity level increases, the corresponding percentage of full disclosure decreases, and the corresponding percentage of null disclosure increases. Note that the percentage of partial disclosure is not shown in the table since it is a dependent random variable calculated by subtracting 100% from the summation of full and null disclosure percentages.

Table 5.5. Random variables for selective data disclosure.

Data Sensitivity Level	Level Value Range (%)	Distribution	Full Disclosure	Null Disclosure
			Value Range (%)	Value Range (%)
1	[0 - 19]	Discrete Uniform	(75 - 90)	(5 - 10)
2	[20 - 39]	Discrete Uniform	(60 - 74)	(11 - 20)
3	[40 - 59]	Discrete Uniform	(31 - 40)	(21 - 30)
4	[60 - 79]	Discrete Uniform	(11 - 20)	(55 - 65)
5	[80 - 100]	Discrete Uniform	(5 - 10)	(75 - 90)

5.4.3. Random Variables for Privacy Policy

Each privacy goal is achieved by enforcing a certain Policy Set (PS). We define random variables for PS processing efforts using uniform distribution with range values 30 to 50 MI (Millions of Instructions).

5.5. System Performance Measures

Two simulation measures are discussed in this subsection, namely *latency,* and *throughput*. For both measures, we define a *job set* as a collection of jobs performed by the system for users. A *job* starts when one or more IPCs generate input data for the job (in the form of one or more

RVS packets) and ends with the delivery of the results to the users—*either* within the delivery of the LRD packet to LUD (for local jobs) *or* within the delivery of the RRD packet to RUD (for remote jobs). In more detail, we define a *single local job* as the complete IPC-EFN-LUD processing-and-transmission sequence, where a single IPC sends one or more RVS packets to EFN, which in turn processes these packets and sends a single LRD packet to a single LUD.[4] Analogously, we define a *single remote job* as the complete IPC-EFN-IFN-Cloud-RUD processing-and-transmission sequence, where a single IPC sends one or more RVS packets to EFN, which then processes these packets and sends one or more ADB packets to IFN, which next processes the received packets and sends one or more ADB packets to Cloud, which finally processes the packets and sends a single RRD packet to a single RUD.[5]

5.5.1. Latency

We define *latency (LAT)* for a job as the period between the moment of submission of a job for execution and the moment when the user receives the complete output for the job. In our case study, LAT is the period from the instant when the *first* bit of RVS for a job is sent by a given IPC till the instant when either the *last* bit of LRD for this job is delivered to LUD or the *last* bit of RRD for this job is delivered to RUD (in this means that we consider the IPC processing time for a job to be negligible).

[4] Note that each single local job can starts with one or more RVS packets, but it should end with a single LRD packet as an output from the EFN processing. We will use the number of LRD packets to count the number of local jobs.

[5] Note that each single remote job can starts with one or more RVS packets, but it should end with a single RRD packet as an output from the Cloud processing. We will use the number of RRD packets to count the number of remote jobs.

LAT is a summation of processing time (PT), queueing time (QT), and transmission time (TT) for a job. The transmission time is the summation of transmission delay (TD) and the propagation delay (PD). Therefore, LAT can be obtained by the following formula [95]:

$$LAT = PT + QT + TT \quad (1)$$

where:

1) PT is a period required for processing a packet by a certain node. It starts when a packet is received by the EFN, IFN, or Cloud node and ends when the packet is put into a transmission queue of the EFN, IFN, or Cloud node, resp. PT for a job includes processing times for its packets. However, in general, is not the sum of these packet processing times since packets of a job can be processed in *parallel* by EFN, IFN, or Cloud. PT is obtained by the following formula [95]:

$$PT = \text{Packet Processing Effort} / \text{Node Processing Speed} \quad (2)$$

2) QD is a period from the moment when a packet is assigned to a queue for transmission till the moment when the packet starts being transmitted. QD depends on the network configuration, node/link parallelism, and the number and frequency of packets generated by the source node. So, QD obtained from the simulation, and it is hard to be calculated.

3) TT is a period from the instant when the *first* bit of the packet is sent (placed on a transmission link) until the instant when the *last* bit of the packet is received by the destination. The transmission time includes two components:

   a) TD is a period from the instant when the first bit is sent (placed on a transmission link) until the instant when the last bit of the packet is sent. TD can be obtained by the following formula:

$$TD = \text{Packet Size} / \text{Link Bandwidth} \quad (3)$$

b) PD is a period from the instant when the last bit of a packet is sent till the instant when the last bit of the packet is received by the destination. PD is an input constant parameter given to the simulation (see Table 5.3).

5.5.2. Throughput

We define *throughput (T)* as the maximum number of jobs that can be completed by a system in a unit of time. To understand ST, we need to explain the following two terms:

1) *Saturation Period* when all system resources (Links and Nodes) are saturated with jobs. We define a *starting saturation point* as the instant when the system becomes saturated which means all system resources are busy with jobs. We also define an *ending saturation point* as the instant when the system becomes unsaturated which means at least one of the system resources becomes idle. Regarding the system saturation points, we recognize three periods:

   a) *Startup Period* from the instance when the *first* bit of RVS is sent by a given IPC until the starting saturation point. This period must be factored out for obtaining the T measure.

   b) *Steady-State Period* from the starting saturation point until the ending saturation point. This is the period of interest to be considered for obtaining the T measure.

   c) *Terminating Period* from the ending saturation point till the instant when either the *last* bit of LRD is delivered to LUD or the *last* bit of RRD is delivered to RUD. The termination period can be optionally considered. In our system, we do not consider the termination period.

2) *Maximum Number of Complete Jobs*—the maximum number of LRD packets delivered to LUDs over IPC-EFN-LUD transmission and processing sequence, or the maximum number of RRD packets delivered to RUDs over IPC-EFN-IFN-Cloud-RUD transmission and processing sequence, respectively.

T measure can be obtained with the following formula:

$$T = \text{Maximum Number of Complete Jobs} / \text{Steady-State Period} \quad (4)$$

where Steady-State Period can be obtained with the following formula:

$$\text{Steady-State Period} = \text{LAT} - \text{Startup Period} \quad (5)$$

The value of Startup Period is determined from the simulation.

5.6. Privacy Control Measures

This subsection discusses measures for selected privacy goals to be achieved by the proposed PEFM system regarding the protection of FHSS data privacy.

5.6.1. Privacy Risk Control

In terms of privacy risk, a system can be either free from any risk or at risk when there are either malicious insider or attacker or both in the system. We define a measure called Privacy Risk Control (PRC) for testing the PEFM performance in controlling privacy risks for different FHSS scenarios (discussed in Section 4.5). For each scenario, we compare the PEFM performance in terms of latency (LAT) and throughput (T) for a scenario with no risk (baseline) with PEFM performance for a scenario with a certain level of privacy risk addressed by PEFM. The comparison results are the cost of privacy control introduced by PEFM. In some cases, this cost is an overhead

added to the system performance. In other cases, this cost is an improvement for the system caused by PEFM.

5.6.2. User Control Level

Privacy policies are classified into several categories; each policy category covers a certain aspect of privacy which is filled by enforcing its policies. For example, data minimization policy is a category of the privacy policy that assures data minimization aspect of privacy. Similarly, we can define data retention policy, selective data disclosure policy, confidential data dissemination policy, and so on. Since privacy policies are specified to add privacy protection level to the sensitive data that in turn specified personal references of a user, enforcing an increasing number of privacy policies increases the level of user control over her data. However, increasing the number of enforced privacy policies will add an extra overhead in terms of system performance (LAT and T). So, in this measure, we need to find out what is the allowed number of privacy policies to be enforced by PEFM and do not impact the system performance to the point of violating the real-time constraints of the system.

We define User Control Level (UCL) as the maximum number of enforced privacy policies for user's sensitive data without violating the real-time constraints for time-sensitive applications with hard-deadlines and with reasonable system performance overhead for not time-sensitive applications with soft-deadlines.

5.6.3. Selective Data Disclosure

We recognize that one of the major PEFM features is its ability to provide different levels of data disclosures according to the level of data sensitivity. This feature needs to be quantitatively

measured in our simulation, so we define a measure called *Selective Disclosure* (SD). For this measure, we consider five numerical levels for data sensitivity 1, 2, 3, 4, 5. The value 1 represents data with the lowest sensitivity level, and the value 5 represents data with the highest sensitivity level. For each sensitivity level, we obtain the rate (percentage) of each of three types of data disclosures named *Full*, *Partial*, and *Null*. We describe the rate of each type of data disclosure as follow:

1) *Rate of Full Data Disclosure* is the percentage of messages that disclose all surveillance data to the authorized LUDs or RUDs.

2) *Rate of Partial Data Disclosure* is the percentage of messages that disclose a part of surveillance data to the authorized LUDs or RUDs.

3) *Rate of Null Data Disclosure* is the percentage of messages that disclose no surveillance data to the authorized LUDs or RUDs.

5.7. Chapter Conclusion

In this chapter, we described our simulation design for the framework of using PEFM for the FHSS system. The discussion of the simulation design included simulation assumptions, simulation parameters, random simulation variables, and the evaluation measures of the simulation. The next chapter will evaluate PEFM's performance based on this simulation design.

CHAPTER 6

PERFORMANCE EVALUATION

6.1. Introduction

This chapter discusses simulation implementation, experimental setup, and the experimental results for the proposed PEFM system. The simulation implementation shows the major simulation components that implement the functionalities of the PEFM system. Different experiments are developed to test the performance of PEFM in different scenarios. Each experiment is built to evaluate PEFM performance in achieving a certain privacy goal and the corresponding system performance results from performing that privacy goal by PEFM.

The main goal of this chapter is to evaluate the feasibility and efficiency of the proposed PEFM solution in protecting sensitive IoT data. To achieve this goal, we discuss the experimental simulation results in terms of the privacy control measures (described in Subsection 5.6) and system performance measures (described in Subsection 5.5). We consider latency and throughput as major system performance measures since they are directly related to the fog and IoT research areas. Latency is so important to be measured to assure that performing privacy control does not break the real-time constraints of the protected system. Throughput is also important to be measured to assure that performing privacy control does not decrease the speedup of the protected system. Since our research investigating privacy protection rather than investigating computer networks, we believe that latency and throughput are good enough for our research as basic

networking measure to be considered for the system evaluation purposes. In addition, we consider response time measure too since it is related to the end user experience. We need to assure that PEFM does not decrease the response time to the point that breaks the hard real-time deadlines. In conclusion, we want to prove the hypothesis that assuring privacy by PEFM should not significantly affect the performance of the protected system.

6.2. Simulation Implementation

The simulation is implemented to evaluate the privacy and performance of the proposed PEFM system in the context of the Foscam Home Surveillance System (FHSS) realistic case study. The FHSS, as well as the framework of using PEFM for FHSS (Section 4.4), are modeled as object-oriented SimPy classes that are used to implement the functionalities of FHSS and PEFM. The simulation includes the following major entities:

1) *Sensor*: A SimPy process class models the entities that act as IoT sensors representing the major data sources in the simulation. Instances of Sensor class are IPC process objects that generate RVS packets. The IPC class object defines a method for generating RVS packets based on a Python generator using the *Generate* process execution method. Each IPC object defines input parameters for RVS packets and determines the starting times for each RVS generation, as specified in Subsection 5.4.1 and Table 5.4.

2) *Node*: A SimPy *resource* class models network nodes that are the processing elements in the simulation. Instances of the Node class are resource objects that represent the processing nodes including EFN, IFN, and Cloud. The attributes of the Node class specify the CPU processing speed of a node and its RAM size, as specified in Table 5.1 in Subsection 5.3.1.

3) *Link*: A SimPy *resource* class models the communication links between the network nodes in the simulation. Instances of the Link class are resource objects that are connecting nodes and carrying packets among them. The attributes of Link class specify the bandwidth, propagation delay and link parallelism, as specified in Table 5.2 in Subsection 5.3.2.

4) *Packet*: A SimPy *process* class models the fundamental unit of communication between nodes in the simulation. Instances of the Packet class are process objects that can represent any type of simulation packet including RVS, LRD, ADB, and RRD. The Packet class defines methods to be performed on a packet object including transmission, processing, and disclosure. The attributes of the Packet class specify packet size, packet processing efforts, data sensitivity level, number of privacy policies to be enforced, and the processing efforts required for each policy, as described in Chapter 5.

5) *Generator*: A SimPy-based Python *Generator* function is used to generate different network configurations in the simulation. The attributes of a Generator function specify EFN fan-in, EFN fan-out, IFN fan-in, Cloud fan-in, and Cloud fan-out, as described in Subsection 5.3.3 and Table 5.3.

6) *Monitor*: A SimPy *monitor* class is used to observe a single variable of interest and to return a data summary either during or at the completion of a simulation run. Instances of the Monitor class are monitor objects. Each one uses the *observe* method to record data on specific variables. We define two monitor objects for each type of packet in the simulation, one for observing starting times (packet generations) and one for observing completion times (packet deliveries). Data collected by these monitor objects are used for calculating the simulation measures including latency, throughput, and response time as specified in Subsection 5.5.

6.3. Experimental Setup

The simulation experiments are conducted to measure addressing privacy risk, assuring selective data disclosure, and increasing user control. We aim to evaluate PEFM system performance in achieving a certain privacy goal. For that, we simulate PEFM for FHSS using SimPy, a process-based discrete event simulation framework based on standard Python, running in the PyCharm, IDE environment. The simulation runs on Intel Core i7-4710 HQ 250 GHz processor with 8 GB RAM. We conducted a series of experiments to measure the overhead or the improvement of using the PEFM for FHSS. For statistical validity, the results are reported based on the mean of data collected over 10 - 20 simulation runs. Since we have random input variables, we try to take the average of multiple runs. The number of runs has chosen arbitrary, but it is within the range where the results become stable. We present and discuss the experimental results in the next section.

We simulate eight different scenarios for using PEFM as a privacy control system to protect the privacy of data in the FHSS (Foscam Home Surveillance System) case study (Section 4.5). We set up and run a simulation for each scenario to measure the PEFM performance in terms of latency (LAT) and throughput (T). These scenarios have different levels of privacy risk of having malicious insiders or attackers for both the local and the remote FHSS applications. A scenario with no risk is considered as a baseline with which a scenario with a certain level of privacy risk is compared. Hence, HNRS and CNRS are the baselines for Local Foscam scenarios (HMIS, HATS, and HHRS) and Remote Foscam scenarios (CMIS, CATS, and CHRS), respectively.

6.4. Experimental Results for Privacy Risk Control

This subsection depicts and discusses the simulation results regarding privacy risk control by PEFM and the corresponding system performance, in terms of latency and throughput, result from this control. The results are first shown and discussed for the local FHSS scenarios using PEFM's LPEM. Then, the results are shown and discussed for the remote FHSS scenarios using PEFM's RPEM.

6.4.1. LPEM Privacy Risk Control and System Performance

Figure 6.1 shows LPEM latency for four local FHSS scenarios: HNRS (baseline), HMIS, HATS, and HHRS. The latency measures for five different network configurations that increase the system load in terms of the number of smart homes (SHs) from 8 SHs up to 200 SHs. Figure 6.2 shows the LPEM latency gain for controlling privacy risk in local FHSS scenarios.

The experimental results, shown in Figure 6.1, show that the LPEM latency for all scenarios slightly increases linearly with configuration complexity. For each network configuration, the LPEM latency for risk scenarios is lower than the baseline which, as expected, is due to data minimization by LPEM. However, using LPEM for controlling privacy risks caused by home insiders results in less latency than using LPEM for controlling privacy risks caused by home attackers. This is due to data minimization by LPEM for the HMIS which is missing for the HATS. For the HATS, data are either fully disclosed to the normal users or null disclosed to the attackers.

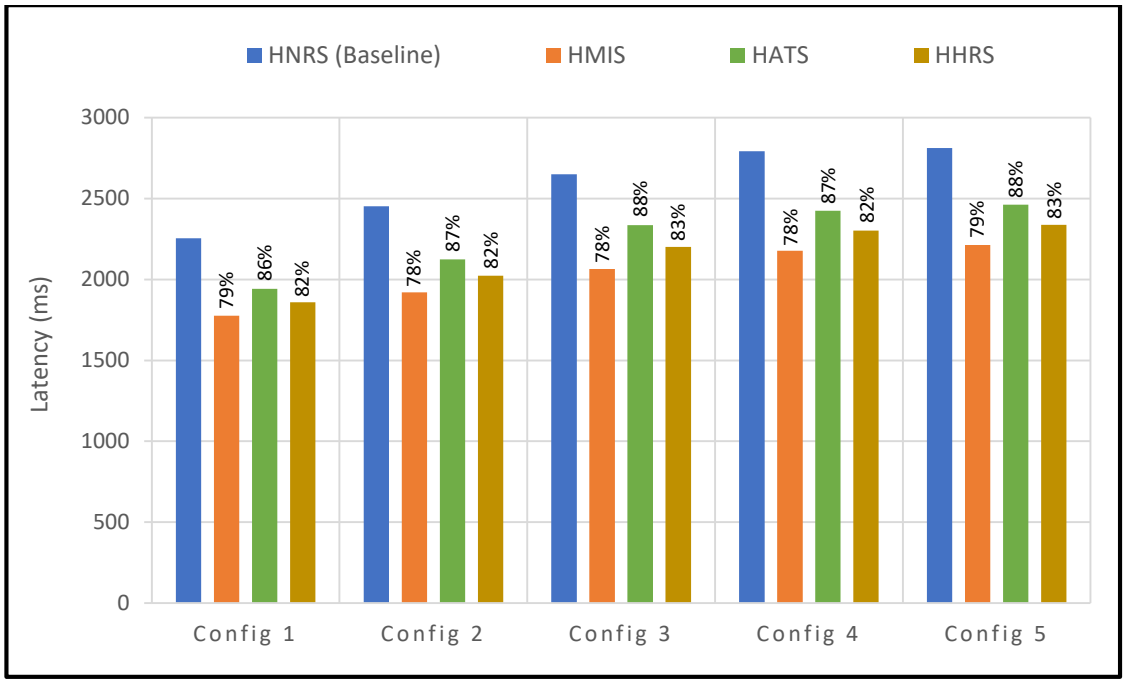

Figure 6.1. Absolute and relative LPEM latency for privacy risk control.

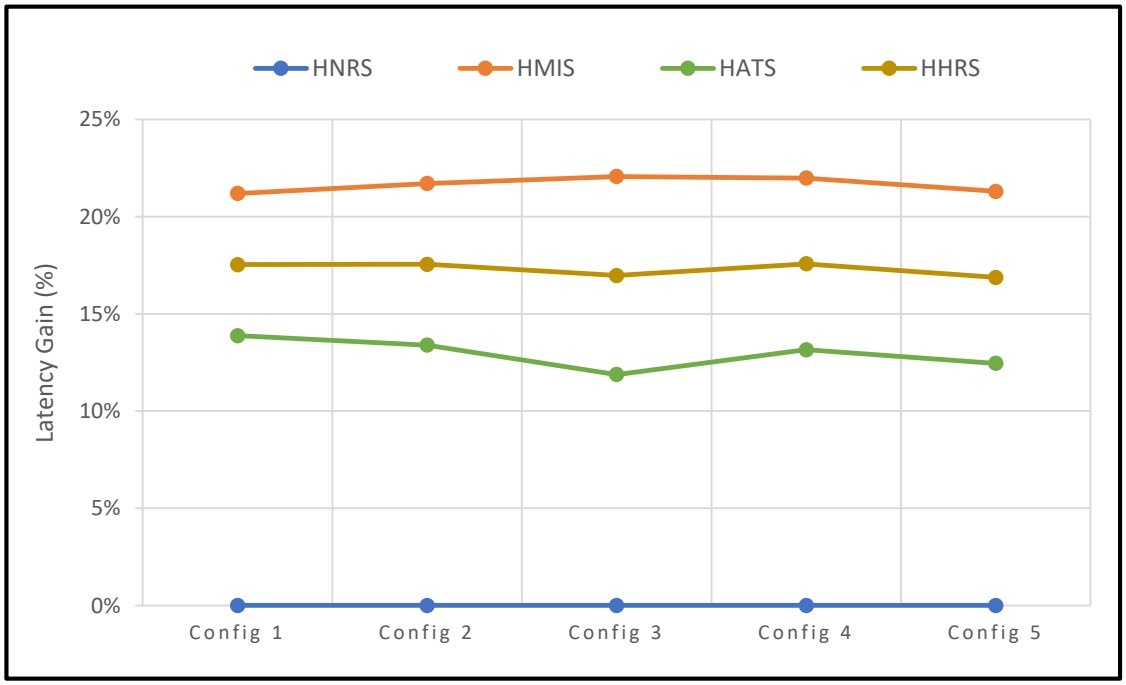

Figure 6.2. LPEM Latency gain for privacy risk control.

Since the ratio of attackers within the local domain is probably low, the amount of fully disclosed data is higher than the amount of null disclosed data. Hence, these fully disclosed data packets take the full time for processing and transmission. For HMIS, most of data packets are partially disclosed to the insiders, which have a higher ratio within local domain than attackers. In this case, LPEM performs data minimization for the partially disclosed data packets in a certain level of disclosure. Hence, the time required for processing and transmitting these minimized data packets is less than what is required for fully disclosed data. For HHRS, that includes both malicious insiders and attackers, as well as the normal users, the LPEM latency is less than the baseline for all configurations.

We observe, even with increasing the system workload, the LPEM latency is not increased proportionally, but its values stay close to each other with only a slight increase. This is due to the distribution feature of fog computing where each SH has an independent fog node (EFN) responsible for processing and transmitting the data of this SH.

Figure 6.2 demonstrates that, compared to the baseline scenario (HNRS), privacy enforcement in LPEM causes an average latency improvement of 21.65% for HMIS, 12.95% for HATS, and 17.30% for HHRS, averaged over five configurations. For each configuration, the graph shows the percentage of LPEM latency gain for this configuration for the three types of scenarios. Among different configurations, the latency gain values are close to each other. However, we observe a dip in the latency gain for *Config 3* (72 SHs) for HATS and a slight dip for HHRS. This is an unexpected case in our results which might be happened due to the randomization of packet size values. Now, among different scenarios, LPEM latency gain is different: higher for HMIS than for HATS. The ratio of LPEM latency gain for HHRS is approximately in the middle between the ratio of gain for each scenario alone.

Since privacy is handled at the local fog node and each SH has a fog node, the complexity should not increase, and our simulation results support that hypothesis. The results show that running LPEM for FHSS improves both privacy and the latency system performance. This is a promising result for the feasibility and efficiency of PEFM in local scenarios.

Figure 6.3 shows LPEM throughput for the local scenarios for five different network configurations. For each scenario type (including the baseline), the LPEM throughput increases more than linearly with configuration complexity. Increasing the number of smart homes (SHs) with *x*-factor increases the throughput with *x*-factor too because more SHs means more output data packets generated. For each network configuration, the LPEM system throughput values are different for each type of scenario. The HMIS increases the system throughput (speed) for all configurations. This is due to the LPEM latency gain for this scenario on the one hand. On the other hand, the LPEM data minimization affects the size of data packets but does not affect the number of packets completed by the system. HATS decreases the system throughput for all configurations. This is because LPEM assures null disclosure for all attackers and hence the number of packets completed by the system is significantly decreased. However, for HHRS, the LPEM system throughput increases for all configurations.

Figure 6.4 demonstrates that, compared to the baseline scenario (HNRS), privacy enforcement in LPEM causes an average throughput *improvement* of 18.45% for HMIS, and causes an average throughput *overhead* of 9.74% for HATS. However, LPEM causes an average throughput *improvement* of 6.06% for HHRS.

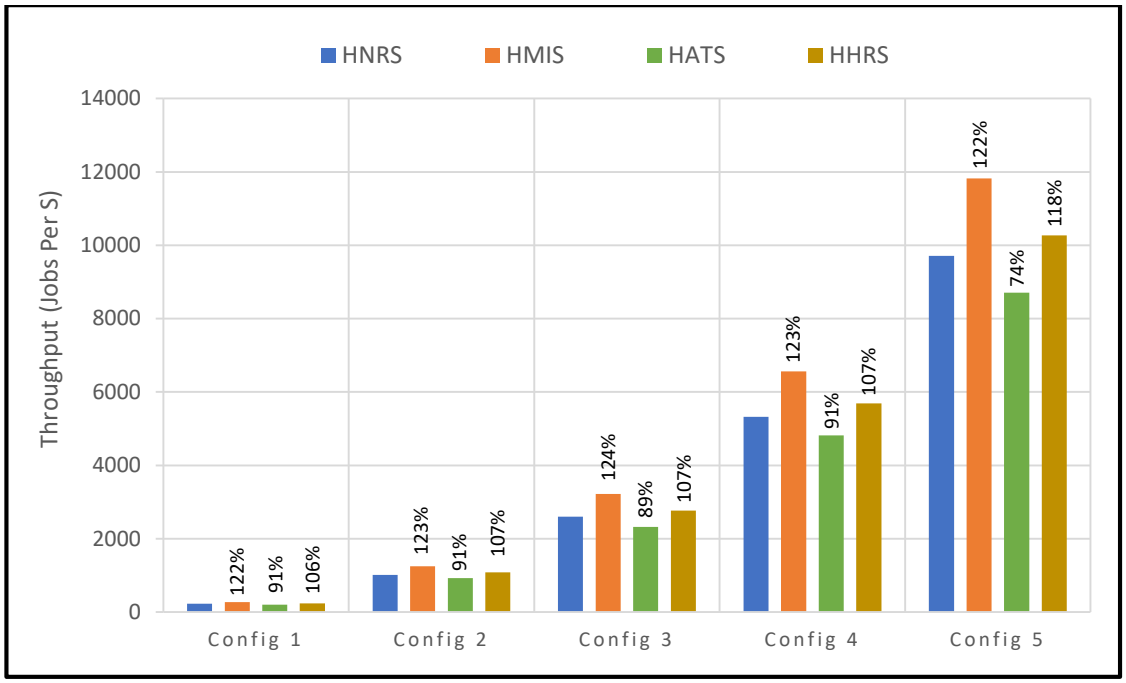

Figure 6.3. Absolute and relative LPEM throughput for privacy risk control.

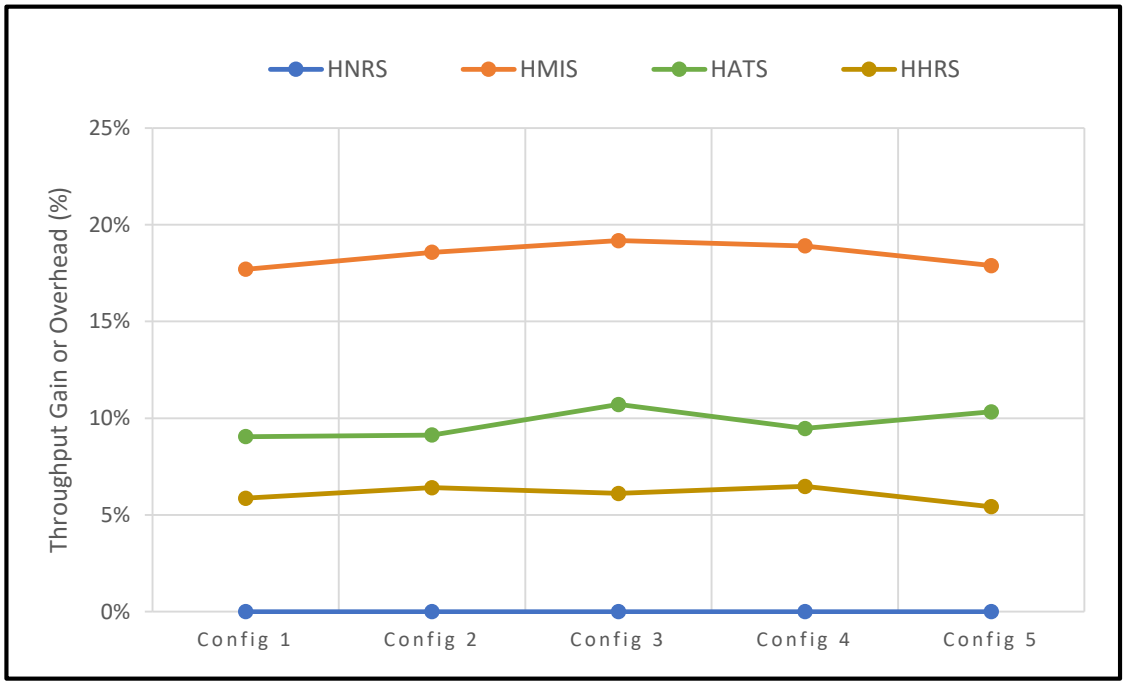

Figure 6.4. LPEM throughput gain and overhead for privacy risk control.

The graph shows that LPEM incurs the highest gain for HMIS. While it incurs overhead instead of gain for HATS. For HHRS, LPEM incurs low throughput gain for the system. This is due to the overhead of HATS.

Regarding system throughput, since privacy is handled at the local fog node and each SH has a fog node, the complexity should not increase, and our simulation results support that hypothesis. The results show that running LPEM for FHSS improves both privacy and the throughput system performance. This is also a promising result for the feasibility and efficiency of PEFM system in local scenarios.

6.4.2. RPEM Privacy Risk Control and System Performance

Figure 6.5 shows RPEM latency for the four remote FHSS scenarios. The latency measures are for five different network configurations. The graph shows that the RPEM latency for all scenario increases linearly with configuration complexity, as expected. For each network configuration, the RPEM latency for risk scenarios is higher than the baseline because controlling privacy risk requires sending data as a form of Active Data Bundle (ADB). Now, each ADB carries the original data in addition to the policies and the policy enforcement engine (ADB's VM). This payload requires more processing and transmission times. This is acceptable in our model since we should have cost for high-level of privacy protection. However, we want to make sure that this cost is reasonable. For HHRS, the RPEM latency is higher than the baseline. This leads us to conclude that for all cases of the remote FHSS system, enforcing privacy policies by RPEM introduces an overhead. This is due that data for the remote system are sent over all stages from IPCs to Cloud among EFN and IFN with less data minimization, than for local scenarios, because the enforcement is done at the final stage from Cloud to RUDs only.

Using RPEM for controlling privacy risks caused by malicious insiders results in slightly higher latency than using RPEM for controlling privacy risks caused by attackers. This is due to the number of packets for CMIS is higher than for CATS, which has a high rate of null disclosures for all attackers in the system. A higher number of packets means more time for processing and transmission; therefore, more system latency.

We observe that increasing the system workload over many configurations results in an observable increase in RPEM latency for remote scenarios. This is different than the case for local scenarios because using Cloud is more centralized than using a distributed fog computing. This highlights the advantage of using fog computing for our solution.

Figure 6.6 demonstrates that, compared to the baseline scenario (CNRS), privacy enforcement in RPEM causes the average latency overhead of 11.51% for CMIS, 11.09% for CATS, and 11.30% for CHRS, averaged over the five configurations.

For each configuration, the graph shows the percentage of RPEM latency overhead for this configuration for the three types of scenarios. The overhead values decrease with configuration complexity; more than linearly for *Config 1* and *Config 2*, and linearly for *Config 3* through *Config 5*. The highest overhead value is for *Config 1*, and the lowest overhead value is for *Config 5*. The reason for this is that when the system workload increases, the computations also increase. Therefore, the time required for enforcing privacy policies by RPEM is proportionally low with respect to the higher computations of *Config 5*. While for *Config 1*, the computations of the system are low so the time for RPEM processing will be more observable than for *Config 5*.

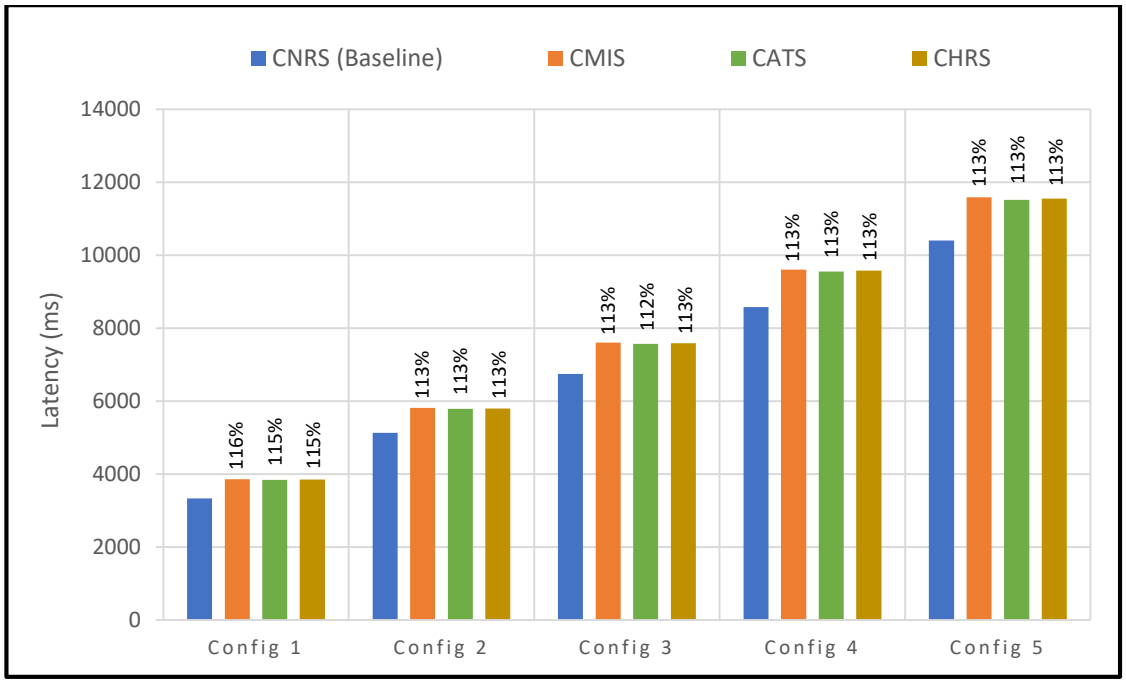

Figure 6.5. Absolute and relative RPEM latency for privacy risk control.

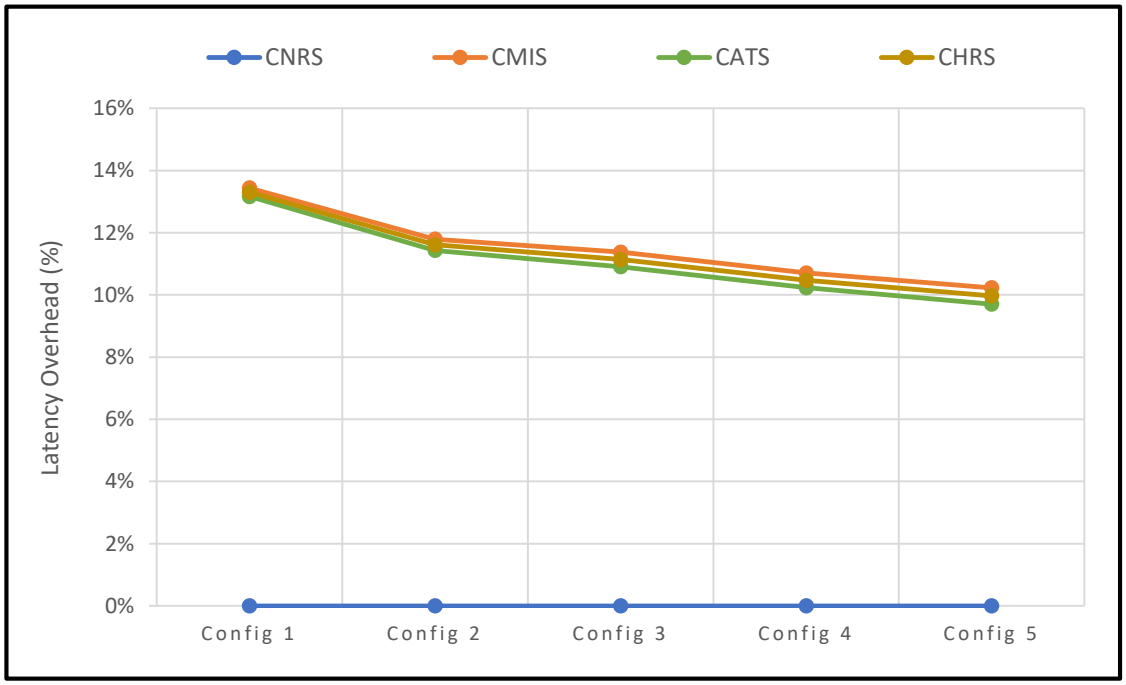

Figure 6.6. RPEM Latency overhead for privacy risk control.

The average system latency overhead (11.3%) due to running privacy enforcement in RPEM is the price paid for privacy protection. So, we have a trade-off of overhead versus the privacy. For some applications, this price might be acceptable while for others it might not. For example, safety applications like fire detection or gas leak detection do not accept any extra overhead. However, entertainment applications might accept the introduced overhead.

Figure 6.7 shows RPEM throughput for the remote scenarios for five network configurations. For each scenario type, the RPEM throughput increases more than linearly with configuration complexity. For each network configuration, the RPEM throughput values are different for each type of scenario. However, RPEM for all scenarios decreases the system throughput (speed) for all configurations. CMIS slightly decreases the throughput since the number of processed packets are the same as for the baseline, but the latency for them is slightly higher. CATS significantly decreases the throughput since the number of processed packets is much less for the baseline due to the null disclosure for all attackers in the scenario.

Figure 6.8 demonstrates that, compared to the baseline scenario (CNRS), privacy enforcement in RPEM causes the average throughput overhead of 15.33% for CMIS, 32.81% for CATS, and 24.07% for CHRS, averages over five configurations. The graph shows that the overhead values are decreased with the configuration complexity; more than linearly for *Config 1* and *Config 2*, and linearly for *Config 3* through *Config 5*. The highest overhead value is for *Config 1*, and the lowest overhead value is for *Config 5*. Among the different configurations, the throughput overhead values are close to each other. However, among different scenarios, RPEM throughput overhead is different: higher for CATS than for CMIS.

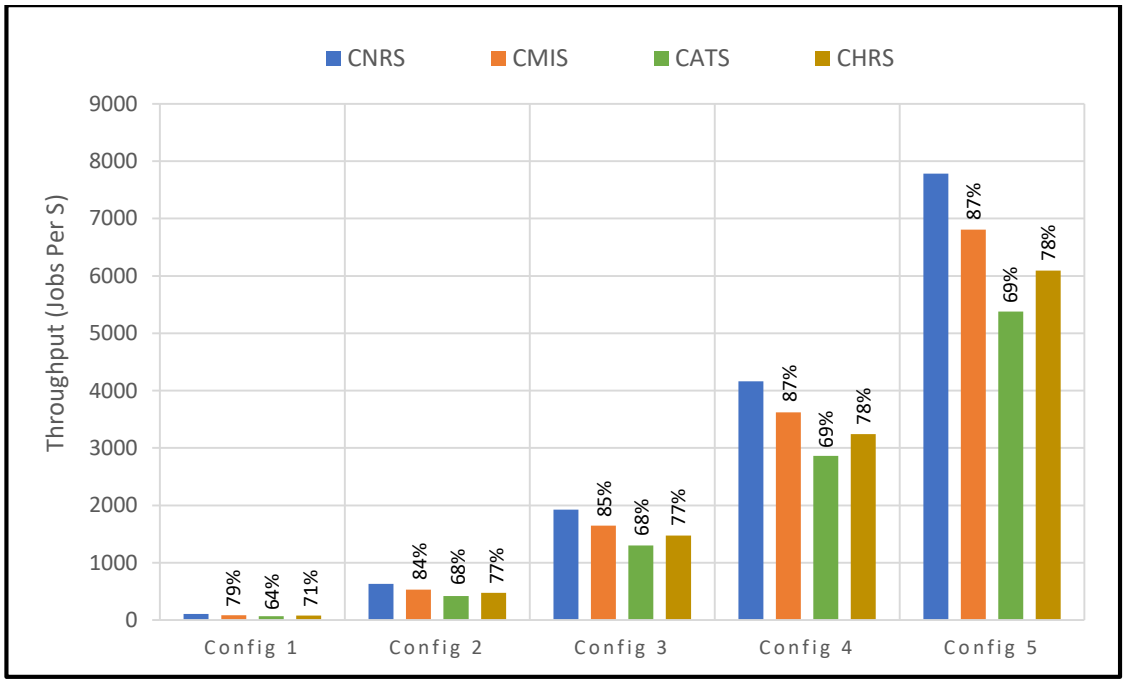

Figure 6.7. Absolute and relative RPEM throughput for privacy risk control.

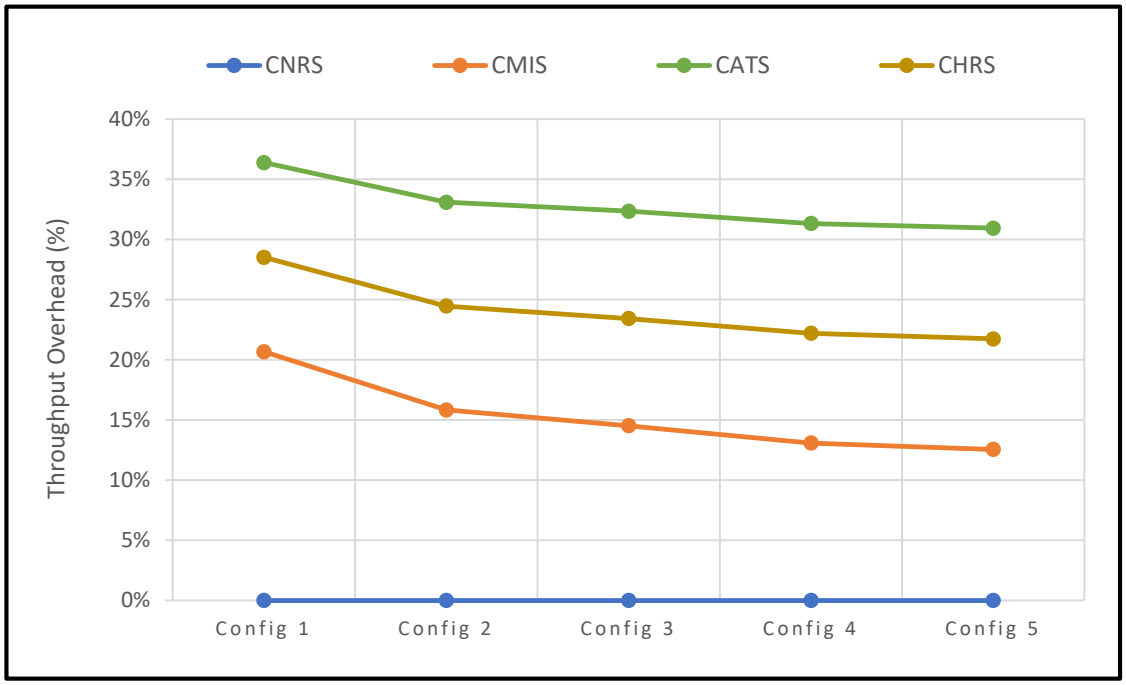

Figure 6.8. RPEM throughput overhead for privacy risk control

The ratio of RPEM throughput overhead for CHRS is approximately in the middle between the ratio of overhead for each individual scenario. The average overhead (24.07%) is the price paid for privacy protection by RPEM.

6.5. Experimental Results for Selective Data Disclosure

This subsection shows and discusses the simulation results regarding selective data disclosure by PEFM and the corresponding system performance, in terms of latency, result from this control. The results are first shown and discussed for the local FHSS scenarios using PEFM's LPEM. Then, the results are shown and discussed for the remote FHSS scenarios using PEFM's RPEM.

6.5.1. LPEM Selective Data Disclosure and System Performance

In this subsection, we show the rate of three data disclosure types: *full*, *partial*, and *null*, for five data sensitivity levels: 1 to 5, over the five configurations. Figure 6.9 shows LPEM selective data disclosure (SDD) for the lowest sensitivity level (1). Figure 6.10 shows LPEM's SDD for moderate sensitivity levels (2-4). Figure 6.11 shows LPEM's SDD for the highest sensitivity level (5). LPEM assures SDD based on data sensitivity levels. For the lowest level, the graph (Figure 6.9) shows, among all configurations, the rate of full data disclosure has the highest values; significantly higher than the rate of partial and null disclosures. The rate of partial disclosure is slightly higher than the rate of null disclosure, which has the lowest rate. The rate of disclosed packets increases with the configuration complexity for all data disclosure types.

For the moderate sensitivity levels, among all configurations, the rate of full data disclosure also has the highest values over partial and null disclosures for sensitivity level = 2. For level 3,

the rate of partial disclosure has the highest values; slightly higher than the rate of full disclosure and significantly higher than the rate of null disclosure. For level 4, the rate of null disclosure has the highest values; higher than the rate of partial and full disclosures. The rate of partial disclosure is higher than the rate of full disclosure, which has the lowest rate.

For the highest level of data sensitivity, LPEM assures SDD with the highest rate of the null disclosure; significantly higher than the rate of partial and full disclosures. The rate of partial disclosure is also higher than the rate of full disclosure, which has the lowest rate.

From all cases above, we observe that when the sensitivity level is increased, the rate of full data disclosure decreases. The rate of partial disclosure also increases with sensitivity level up to 3 (middle level) then it decreases with the upper levels.

Figure 6.12 demonstrates that, compared to the baseline, LPEM's SDD causes an average latency improvement of 9.05% for level 1, 17.65% for level 2, 31.50% for level 3, 44.52% for level 4, and 54.87% for level 5, over five configurations. This means the ratio of latency gain increases with the sensitivity level. This is because the corresponding ratio of data minimization increases and hence the latency decreases.

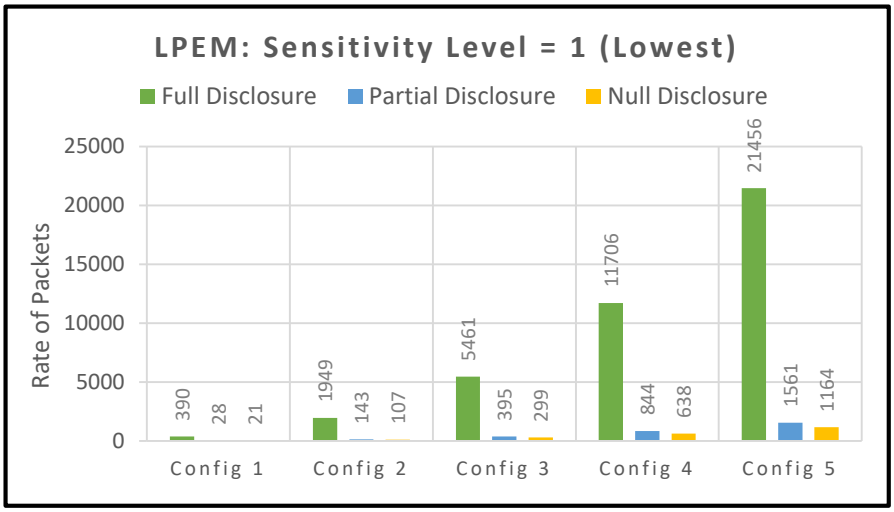

Figure 6.9. LPEM selective data disclosures for the lowest sensitivity level.

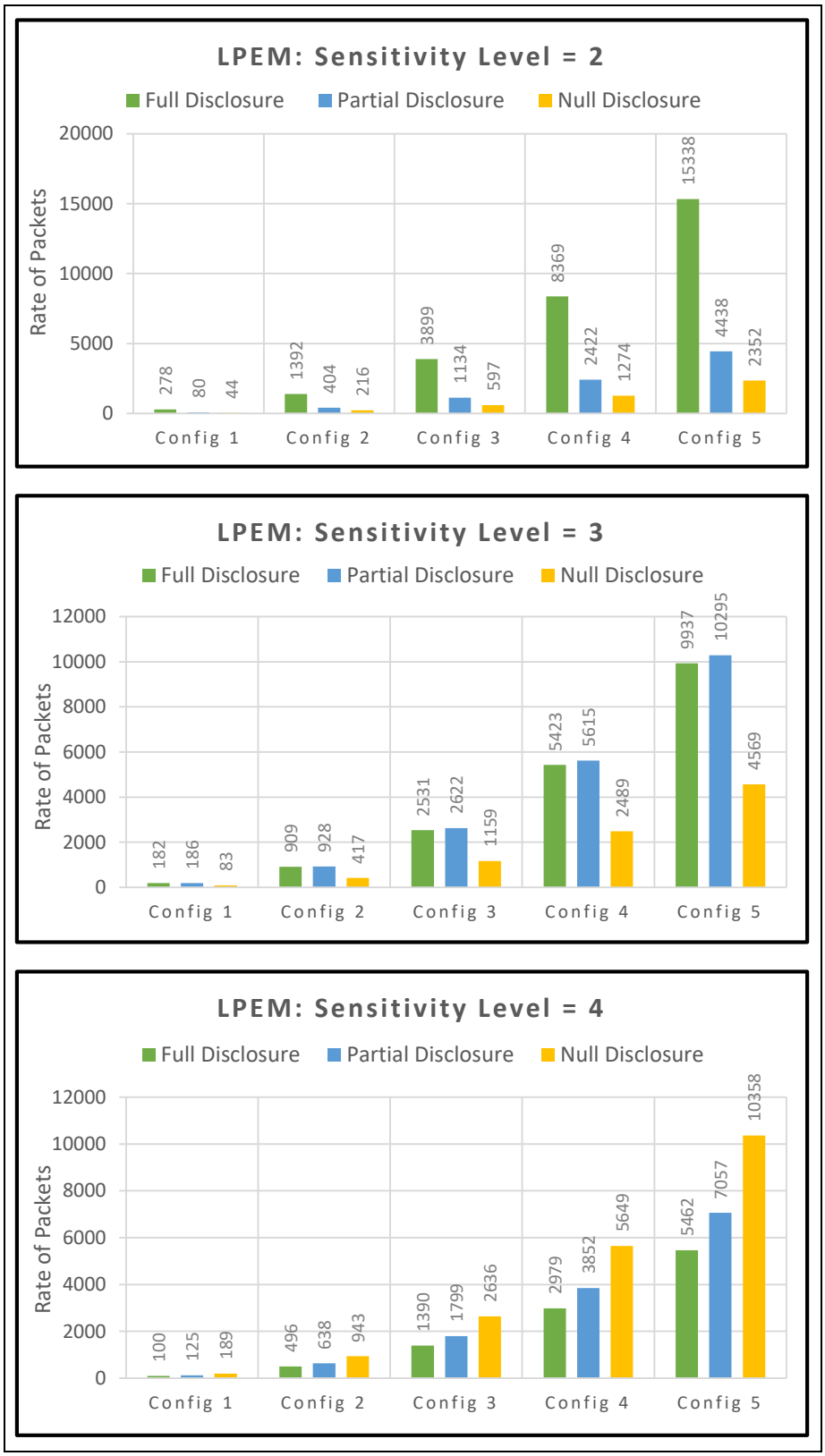

Figure 6.10. LPEM selective data disclosures for moderate sensitivity levels.

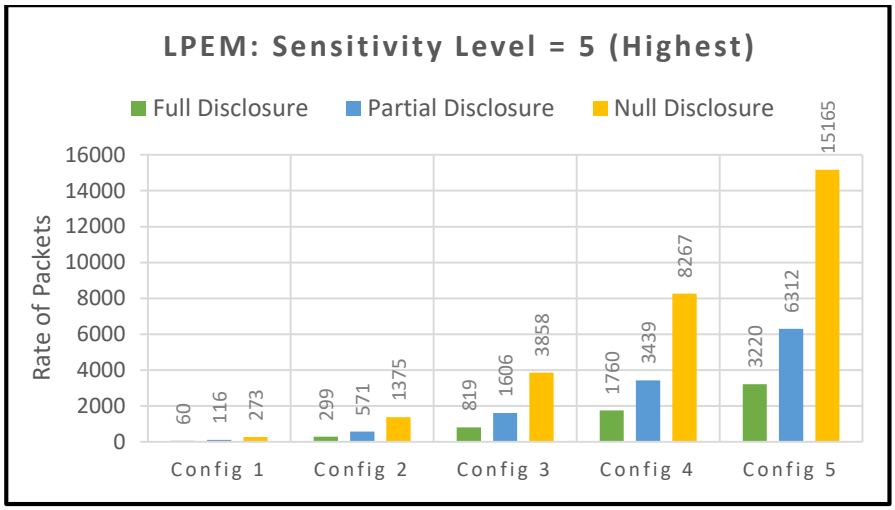

Figure 6.11. LPEM selective data disclosures for the highest sensitivity level.

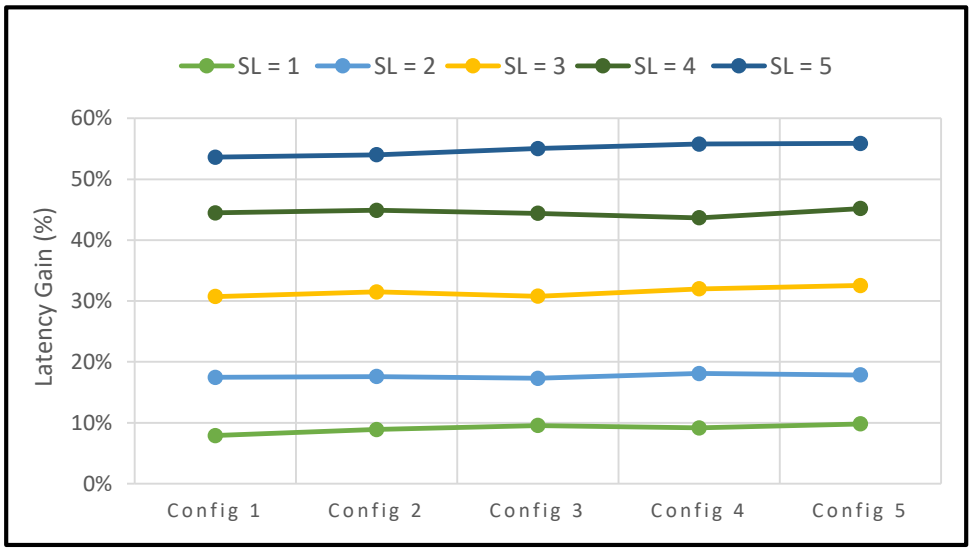

Figure 6.12. LPEM latency gain by selective data disclosure.

6.5.2. RPEM Selective Data Disclosure and System Performance

In this subsection, we also show the rate of three data disclosure types: full, partial, and null, for five data sensitivity levels: 1 to 5, over five configurations. Figure 6.13 shows RPEM selective data disclosure (SDD) for the lowest sensitivity level (1). Figure 6.14 shows RPEM's SDD for moderate sensitivity levels (2-4). Figure 6.15 shows RPEM's SDD for the highest sensitivity level (5).

RPEM assures SDD based on data sensitivity levels. For the lowest level, the graph shows, among all configurations, the rate of full data disclosure has the highest values; significantly higher than the rate of partial and null disclosures. The rate of partial disclosure is slightly higher than the rate of null disclosure, which has the lowest rate. The rate of disclosed packets increases with the configuration complexity for all data disclosure types.

For the moderate sensitivity levels, among all configurations, the rate of full data disclosure also has the highest values over partial and null disclosures for sensitivity level = 2. For level 3, the rate of partial disclosure has the highest values; slightly higher than the rate of full disclosure and significantly higher than the rate of null disclosure. For level 4, the rate of null disclosure has the highest values; higher than the rate of partial and full disclosures. The rate of partial disclosure is higher than the rate of full disclosure, which has the lowest rate.

For the highest level of data sensitivity, RPEM assures SDD has the highest rate of the null disclosure; significantly higher than the rate of partial and full disclosures. The rate of partial disclosure is also higher than the rate of full disclosure, which has the lowest rate.

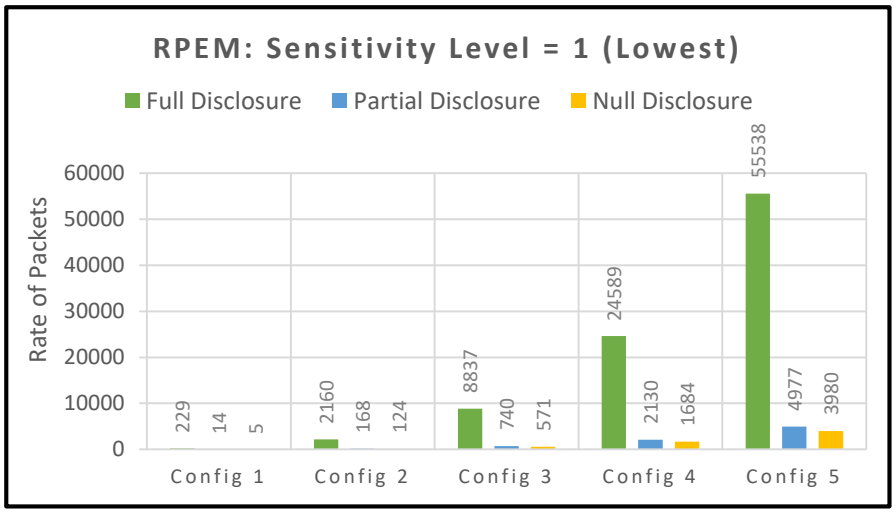

Figure 6.13. RPEM selective data disclosures for the lowest sensitivity level.

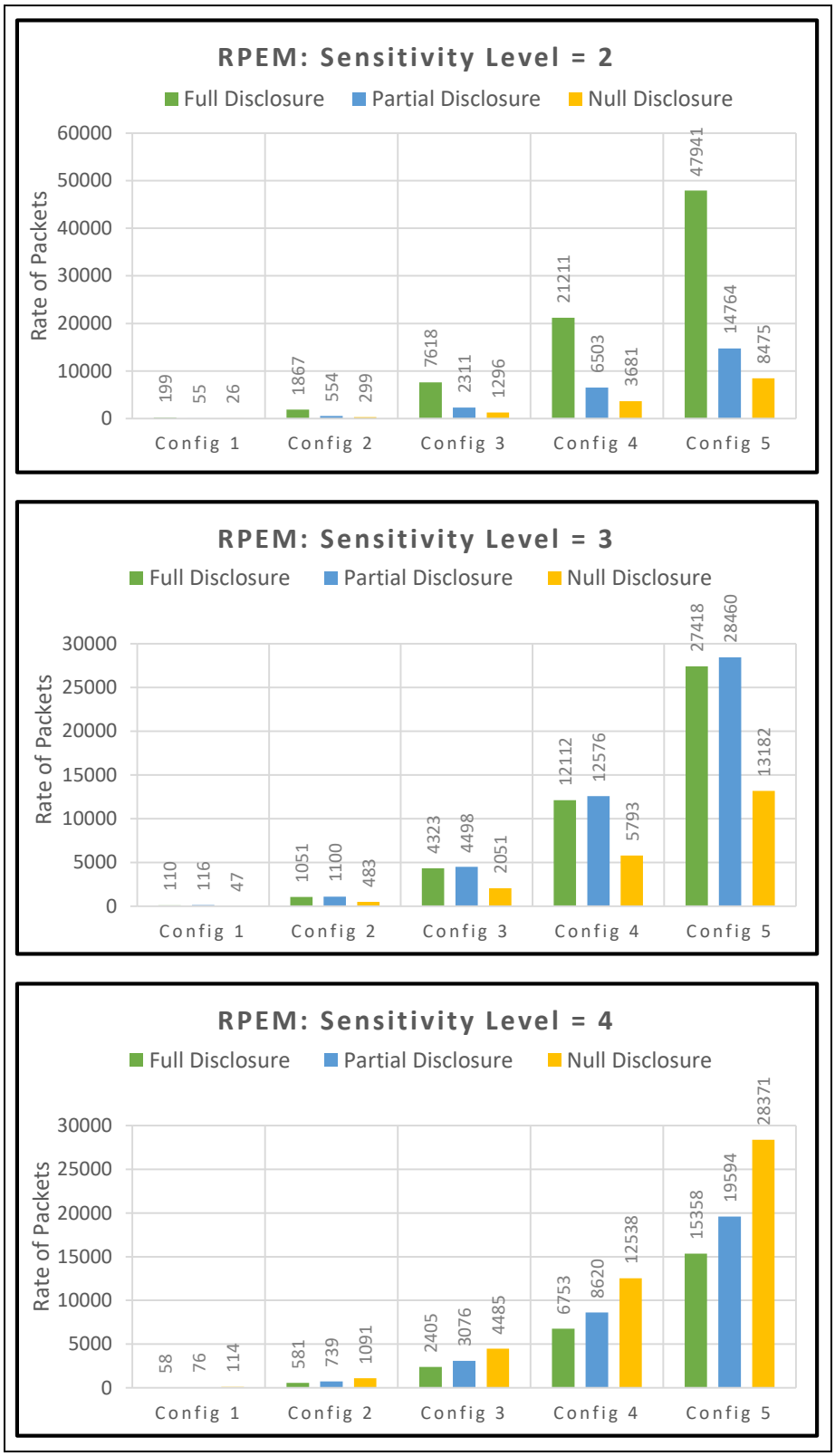

Figure 6.14. RPEM selective data disclosures for moderate sensitivity levels.

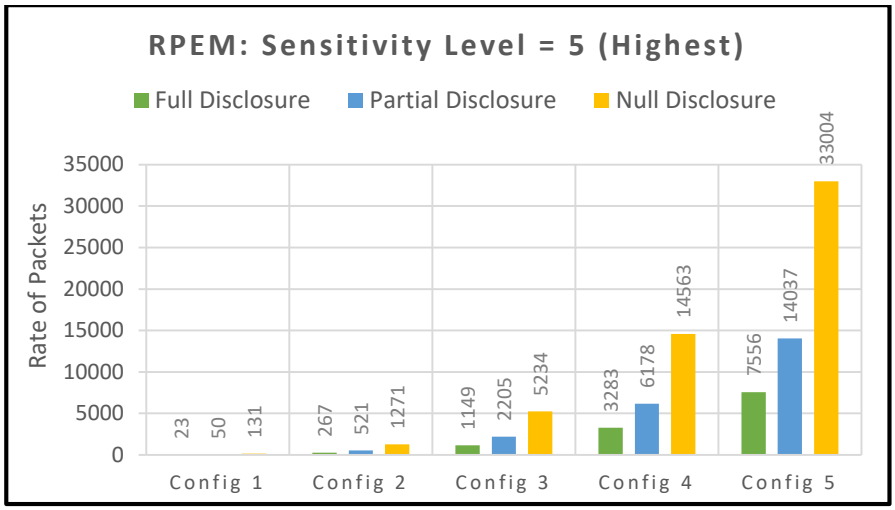

Figure 6.15. RPEM selective data disclosures for the highest sensitivity level.

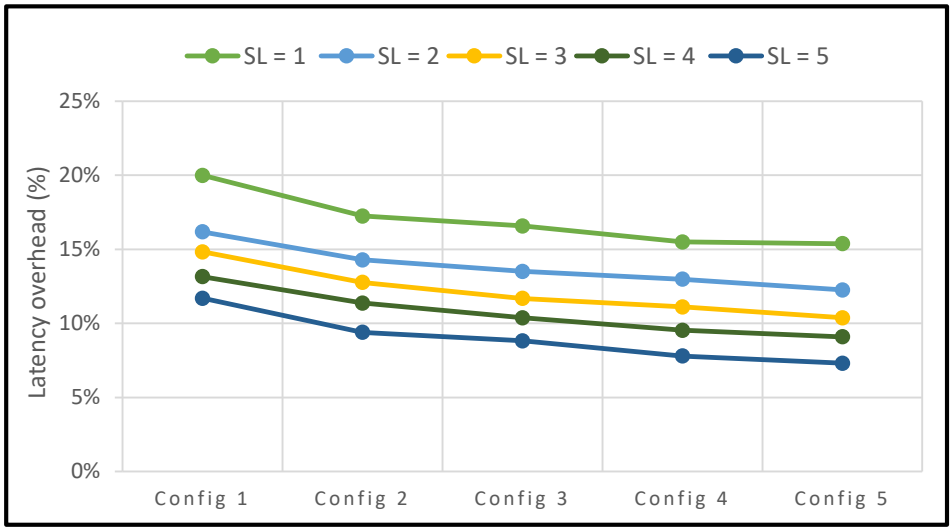

Figure 6.16. RPEM latency overhead by selective data disclosure.

From all cases above, we observe that when the sensitivity level increases, the rate of full data disclosure decreases. The rate of partial disclosure also increases with sensitivity level up to 3 (middle level) then it decreases with the upper levels.

Figure 6.16 demonstrates that, compared to the baseline, RPEM's SDD causes the average latency overhead of 16.94% for level 1, 13.84% for level 2, 12.15% for level 3, 10.71% for level 4, and 9.00% for level 5, over five configurations. This means the ratio of latency overhead

decreases with the sensitivity level. This is because the data minimization is done only at the final stage of transmission between Cloud and RUDs.

6.6. Experimental Results for User Control

This subsection shows and discusses the simulation results regarding PEFM's ability to increase user control by enforcing an increased number of user privacy policies and the corresponding system performance, in terms of response time. The results are first shown and discussed for the local FHSS using PEFM's LPEM. Then, the results are shown and discussed for the remote FHSS using PEFM's RPEM.

6.6.1. LPEM User Control and System Performance

To assure that increasing the number of enforced privacy policies does not break the real-time constraints, we need to determine the hard-real-time (HRT) deadline and the soft-real-time (SRT) deadline of the system. These real-time deadlines are considered for the response time which is the most important measure for real-time IoT applications.

For that, we run 30 simulation iterations for the LPEM's baseline case and for each iteration we measure the system response time, as shown in Figure 6.17 below. The results show that the minimum response time, among all iterations, is 51 ms which is considered as the HRT of the LPEM system. The results also show that the maximum response time, among all iterations, is 126 ms which is considered as the SRT of the LPEM system. Therefore, we identify the interval [51–126] as the real-time interval for the LPEM system. Any response time within the real-time interval is acceptable for real-time FHSS applications.

Increasing the number of enforced privacy policies by LPEM from 1 to 20 results in response times from 48 to 84 ms which is within the real-time interval, as shown in Figure 6.18. According to that, LPEM does not break the real-time constraints even with an increased number of policies. This is mainly due to LPEM's data minimization, as explained before. This result shows that for applications with HRT deadline, enforcing 1 to 5 policies is acceptable. For applications with SRT deadline, enforcing 6-20 policies is acceptable.

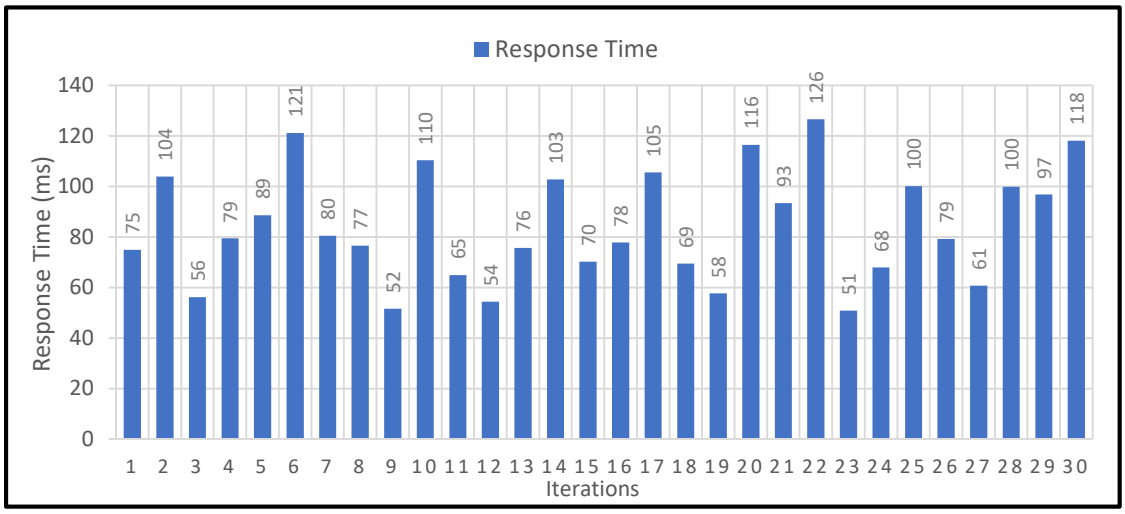

Figure 6.17. LPEM's hard real-time and soft real-time deadlines for response times.

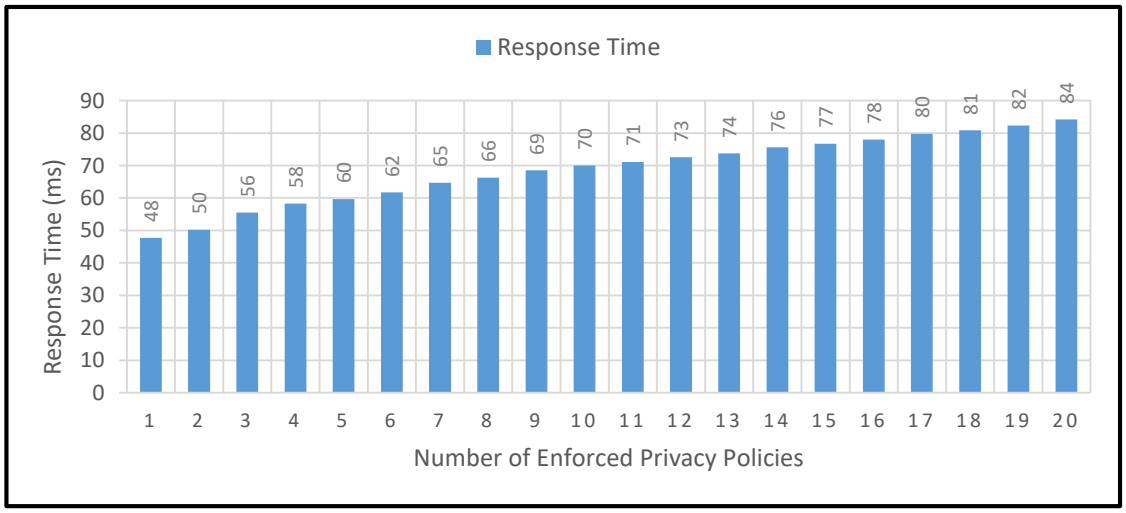

Figure 6.18. LPEM response times for increasing user control.

6.6.2. RPEM User Control and System Performance

Again, to assure that increasing the number of enforced privacy policies by RPEM does not break the real-time constraints of the remote FHSS system, we need to determine the HRT and SRT deadlines of the system. For that, we run also 30 simulation iterations for the RPEM's baseline case and for each iteration we measure the system response time, as shown in Figure 6.19 below.

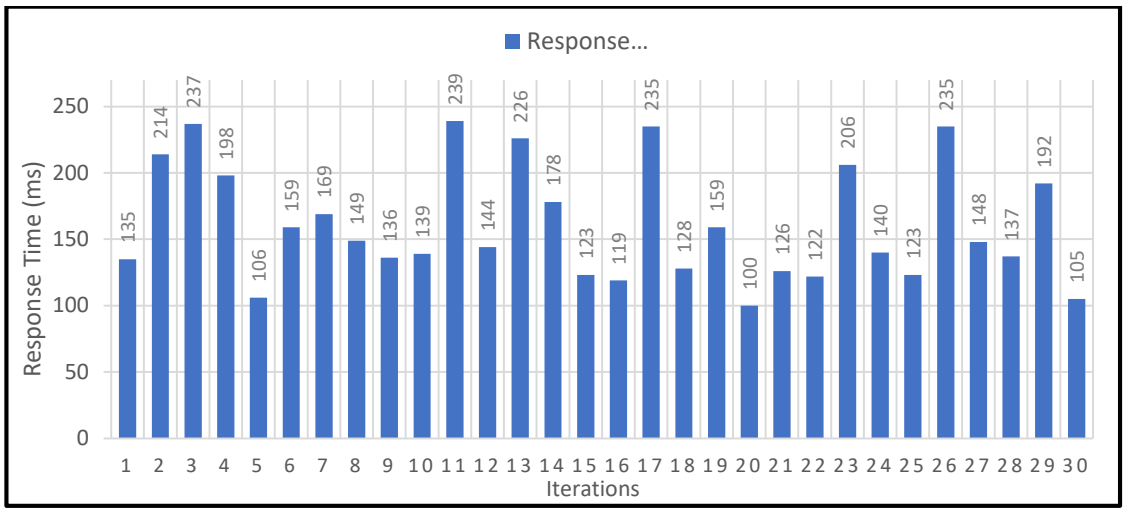

Figure 6.19. RPEM's hard real-time and soft real-time deadlines for response times.

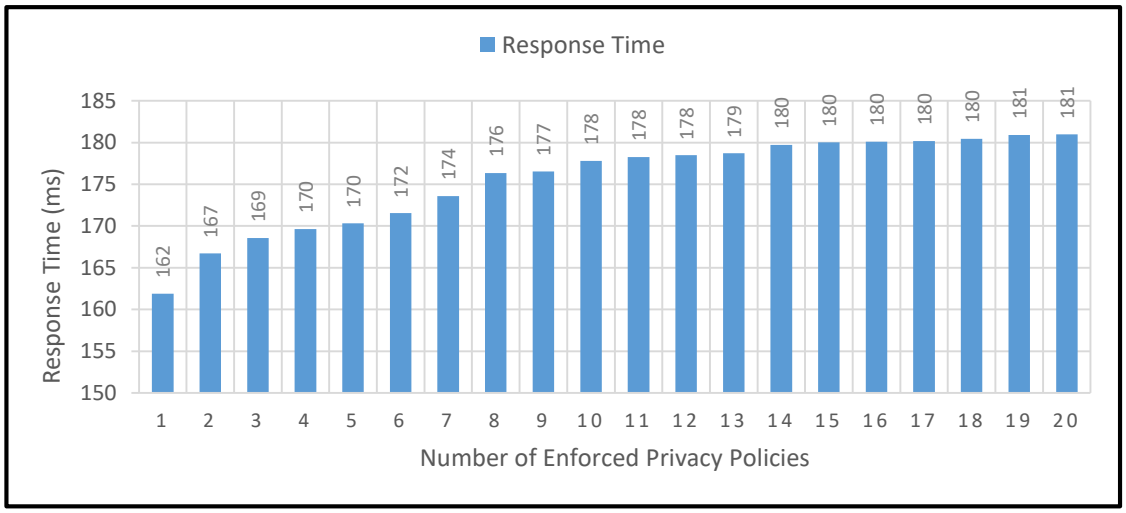

Figure 6.20. RPEM response times for increasing user control.

The results show that the minimum response time, among all iterations, is 100 ms which is considered as the HRT of the RPEM system. The results also show that the maximum response time, among all iterations, is 239 ms which is considered as the SRT of the RPEM system. Therefore, we identify the interval [100–239] as the real-time interval for the RPEM system. Any response time within the real-time interval is acceptable for remote real-time FHSS applications.

Increasing the number of enforced privacy policies by RPEM from 1 to 20 results in response times from 162 to 181 ms, as shown in Figure 6.20. The minimum response time is higher than the RTD due to the latency for transmitting and enforcing privacy policies by Active Data Bundles (ADBs). However, the maximum response time is much less than the SRT due to data minimization is done by RPEM's ADBs because of evaporation and apoptosis. According to that, RPEM works within the real-time interval even with an increased number of policies.

This result shows that for applications with real-time deadlines higher than the HRT deadline, enforcing 1 to 5 policies, is acceptable. For applications with deadlines less than or equal the SRT deadline, enforcing 6-20 policies is acceptable.

6.7. Chapter Conclusion

In this chapter, we discussed the simulation implementation that shows the major simulation components for implementing the functionalities of the proposed PEFM system as a privacy control for the FHSS. Then, the chapter presented the experimental setup for PEFM in a different scenario to enable testing its performance in different situations and for different evaluation measures. Lastly, the chapter presents the simulation results for different privacy measures and system performance measures. The results show that PEFM is feasible and efficient especially when most of the work is done in the fog domain.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1. Conclusions

In this dissertation, we have proposed a solution, called the Policy Enforcement Fog Module (PEFM), for protecting sensitive IoT data throughout their entire lifecycle. The proposed module protects sensitive IoT data from unauthorized disclosures and limits authorized access to these data based on a need-to-know basis only. PEFM has enhanced fog computing by adding to it a mechanism for supporting data privacy. Furthermore, the proposed PEFM system extends privacy protection for fog data to non-fog data using the power of active data bundles. It can serve both local IoT applications via the Local Policy Enforcement Module (LPEM) and remote IoT applications via Remote Policy Enforcement Module (RPEM) for both real-time and non-real-time IoT applications.

This study is limited to protecting private (sensitive) data only; public IoT data are outside of the research scope. We have relied on the data owner to identify which data are sensitive, so identifying sensitive data is also outside of the research scope. The research has focused on assuring the privacy of data rather than other aspects of privacy such as user privacy, context privacy, and so on. Privacy has been the major focus, so security and trust are also outside of the scope of this research. However, some security and trust aspects that serve privacy are considered. Lastly, we

have limited our investigation to fog computing rather than any similar terminologies—including Cloudlets, Mobile Edge Computing , Mobile Cloud Computing , and so on.

We have developed a high-level conceptual design for the proposed PEFM system illustrating its structural components, and their interactions with other IoT entities. Then, we have presented a system design for the major PEFM components, namely LPEM and RPEM, outlining their structure and the process flow of the interactions among their structural components and the IoT environment. A flow diagram showing the algorithm for each PEFM's module (LPEM or RPEM) functionalities are also presented.

To test the performance of PEFM in a realistic IoT environment, we have introduced a proof-of-concept case study named Foscam Home Surveillance System (FHSS) and have described its structure and its devices' technical specifications. Then, we have identified privacy threats in the existing FHSS case study and presented the framework for using the PEFM system as a privacy control for FHSS to address the identified threats. Lastly, we have discussed eight scenarios for the FHSS case study in terms of privacy risk levels for both local and remote FHSS data processing.

To set up an evaluation environment for PEFM, we have developed a comprehensive simulation design for the framework of using PEFM for the FHSS system. The design has included simulation assumptions, simulation parameters, random simulation variables, and the evaluation measures of the simulation.

Finally, we have implemented a simulation for the proposed PEFM system as a privacy control for FHSS. The simulation has run many experiments for different scenarios to enable testing its performance in different situations and for different evaluation measures. Simulation experiments have been implemented using SimPy, a process-based discrete-event simulation framework based on standard Python. The experimental results have been discussed in terms of the

privacy goals achieved by PEFM and the corresponding system performance overhead or gain introduced in terms of latency, throughput, and response time.

The results have shown that PEFM's LPEM (local system) controls privacy risks of having malicious insiders or attackers or both of them in the system with a 17.3% average latency gain, compared with the local baseline system with no risk. In terms of system throughput, controlling privacy risk by LPEM has introduced an 18.45% average throughput gain when having insiders; a 9.74% average throughput gain when having attackers; and a 6.06% average throughput gain when having both. While for the PEFM's RPEM (remote system), controlling privacy risks when having malicious insiders or attackers or both of them in the system, RPEM has introduced an 11.3% average latency overhead and a 24.07% average throughput overhead, compared with the remote baseline system with no risk. For privacy risk control, we have concluded that PEFM has improved the privacy and the system performance as well for the local FHSS system. The gain in terms of latency and throughput is mainly due to data minimization. While, for the remote FHSS system, PEFM has improved the privacy with a reasonable overhead in system performance. This overhead is the price to be paid for a higher level of privacy in terms of lifecycle data protection. The results have also shown that PEFM assures selective data disclosure based on data sensitivity with a 31.51 average latency gain by LPEM and a 12.52% average latency overhead by RPEM, among five data sensitivity levels. So, there is a high gain in the system performance for local FHSS and a reasonable overhead in system performance for the remote FHSS. Finally, the results have shown that PEFM increases users' control for their data with the number of enforced privacy policies. For a local system, LPEM increases user control by enforcing 1 to 20 privacy policies with only 48 to 84 milliseconds average response time. This response time interval has been within the local real-time interval, which is identified as [51– 126] milliseconds, where the 51 has considered as a hard real-

time deadline and 126 as a soft real-time deadline. For the remote system, RPEM increases user control by enforcing 1 to 20 privacy policies with 162 to 181 milliseconds average response time. This response time interval has been within the remote real-time interval, which is identified as [100– 239] milliseconds, where the 100ms is a hard real-time deadline, and 239ms is a soft real-time deadline.

From all results, we have concluded that better privacy controls with minimal overhead can be achieved if most PEFM processes are executed by the local fog nodes. Migrating parts of PEFM processes to remote fog nodes or the cloud incurs more overhead than using strictly local fog nodes. This overhead is the price to be paid for a higher level of privacy in terms of lifecycle data protection. So, there is a tradeoff between overhead and the desired level of privacy. The overhead should be acceptable by applications that are not time-sensitive with hard deadlines.

7.2. Future Work

Doing this dissertation work gives us the knowledge and experience to improve the current work in the following major directions:

1) Investigating privacy policy negotiation for IoT toward enabling data owners to specify and control the privacy policy for their personal data.
2) Using machine learning techniques for identifying sensitive data by fog nodes that host instances of the PEFM solution.
3) Developing lightweight fog-based active bundles for time-sensitive IoT applications with hard deadlines.
4) Evaluating the PEFM performance in different IoT application areas such as connected vehicles, connected industry, smart city, and smart energy.

BIBLIOGRAPHY

[1] G. Yang, J. Xu, W. Chen, H. Z. Qi, and H. Y. Wang, "Security characteristic and technology in the Internet of Things," *The Academic J. of Nanjing University of Posts and Telecommunications (Natural Science)*, vol. 30(4), 2010, pp 20–29.

[2] J. Green, B. McCarson, and M. Devine, "Building the Internet of Things," (Slides), *Internet of Things World Forum*, Chicago, IL, 2014. Accessed on March 18, 2017, from: https://daue6ehqissah.cloudfront.net/breakou

[3] S. Kraijak and P. Tuwanut, "A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends," *11th IEEE Intl. Conf. on Wireless Communications, Networking and Mobile Computing (WiCOM)*, Shanghai, China, Sep. 2015, pp 1–6.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17(4), 2015, pp 2347–2376.

[5] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, Elsevier, vol. 29(7), Sep. 2013, pp 1645–1660.

[6] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Trans. on Industrial Informatics*, vol. 10(4), Nov. 2014, pp 2233–2243.

[7] Cisco, "Fog computing and the Internet of Things: Extend the cloud to where the things are," *White paper*, San Jose, CA, Accessed on March 26, 2017, from: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[8] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things J.*, vol. 3(5), Oct. 2016, pp 637–646.

[9] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *IEEE*, vol. 103(1), Jan. 2015, pp 14–76.

[10] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, “Network function virtualization: Challenges and opportunities for innovations,” *IEEE Communications Magazine*, vol. 53(2), Feb. 2015, pp 90–97.

[11] M. Zanjireh and H. Larijani, “A survey on centralized and distributed clustering routing algorithms for WSNs,” *81st IEEE Vehicular Technology Conf. (VTC Spring)*, Glasgow, Scotland, May 2015, pp 1–6.

[12] A. Ikram, A. Anjum, R. Hill, N. Antonopoulos, L. Liu, and S. Sotiriadis, “Approaching the Internet of Things (IoT): a modelling, analysis and abstraction framework,” *Concurrency and Computation: Practice and Experience*, vol. 27(8), June 2015, pp 1966–1984.

[13] Internet Society, “Policy brief: The Internet of Things,” *Briefing Papers*, Oct. 2016. Accessed on May 15, 2017, from: http://www.internetsociety.org/policybriefs/iot

[14] L. Ben Othmane, “Active bundles for protecting confidentiality of sensitive data throughout their lifecycle,” Ph.D. Dissertation, Department of Computer Science, Western Michigan University, Kalamazoo, MI, Dec. 2010.

[15] N. Lord and C. Davis, “An expert guide to securing sensitive data: 34 experts reveal the biggest mistakes companies make with data security,” *Digital Guardian*, 2017. Accessed on May 15, 2017, from: https://digitalguardian.com/blog/expert-guide-securing-sensitive-data-34-experts-reveal-biggest-mistakes-companies-make-data

[16] K. Barker, M. Askari, M. Banerjee, K. Ghazinour, B. Mackas, M. Majedi, ... and A. Williams, “A data privacy taxonomy,” *British National Conf. on Databases, Dataspace: The Final Frontier, BNCOD, Lecture Notes in Computer Science (LNCS)*, vol. 5588, Springer, Berlin, Germany, 2009, pp 42–54.

[17] A. Al-Hasnawi, A. Al-Gburi, and L. Lilien, “Protecting sensitive data in IoT using active data bundles,” *Trans. of the 12th Western Michigan IT Forum, Big Data Analytics & Internet of Things (IoT)*, Kalamazoo, MI, Nov. 2016, pp. 17–27.

[18] H. T. Reis, and C. R. Knee, “What we know, what we don’t know, and what we need to know about relationship knowledge structures,” *Knowledge Structures in Close Relationships*, 1996, pp 169–91.

[19] H. Ziegeldorf, G. Morchon, and K. Wehrle, "Privacy in the Internet of Things: Threats and challenges," *Security and Communication Networks*, vol. 7(12), Dec. 2014, pp 2728–2742.

[20] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management," Version v0.34, Aug. 2010.

[21] S. Pearson, "Taking account of privacy when designing cloud computing services," *ICSE W. on Software Engineering Challenges for Cloud Computing*, Vancouver, Canada, 2009, pp 44–52.

[22] A. Al-Hasnawi and L. Lilien L, "Privacy services and mechanisms," (Slides), Department of Computer Science, Western Michigan University, Kalamazoo, MI, Jan. 2016.

[23] S. Poslad, "Ubiquitous computing: smart devices, environments and interactions," *John Wiley & Sons*, Aug. 2011.

[24] L. Ben Othmane and L. Lilien, "Protecting privacy in sensitive data dissemination with active bundles," *Seventh Annual Conf. on Privacy, Security and Trust (PST)*, Saint John, New Brunswick, Canada, Aug. 2009, pp 202–213

[25] C. Murphy, "The need to secure sensitive data in Hadoop and IoT ecosystems," *HPE Security – Data Security*, 2017. Accessed on May 25, 2017, from: https://www.voltage.com/iot/need-secure-sensitive-data-hadoop-iot-ecosystems/

[26] S. Viala, "Internet of Things: New challenges and practices for information governance," 2015. Accessed on May 25, 2017, from: http://www.revasolutions.com/internet-of-things-new-challenges-and-practices-for-information-governance/

[27] "Trends in encryption and data security" (Slides), Cloud, Big Data and IoT Edition, *Vormetric Data Threat Report, Vormetric Data Security*, San Jose, CA, 2016.

[28] M. Alani, "Elements of cloud computing security: a survey of key practicalities," *Springer Briefs in Computer Science*, 2016.

[29] A. Al-Gburi, A. Al-Hasnawi, and L. Lilien, "Differentiating security from privacy in Internet of Things—A survey of selected threats and controls," Chapter 9 in: K. Daimi et al., *Computer and Network Security Essentials*, Springer, 2018, pp. 153–172.

[30] M. Satyanarayanan, P. Bahl, R. Caceres, and N Davies, “The case for VM-based cloudlets in mobile computing,” *IEEE Pervasive Computing*, vol. 8(4), Oct.–Dec. 2009, pp 14–23.

[31] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile edge computing—A key technology towards 5G,” White Paper, *European Telecommunications Standards Institute*, Sophia-Antipolis, France, No. 11, Sep. 2015.

[32] N. Fernando, S. W. Loke, and W. Rahayu, “Mobile cloud computing: A survey,” *Future Generation Computer Systems*,” Elsevier, vol. 29(1), Jan. 2013, pp 84–106.

[33] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D.E. Culler, “SPINS: Security protocols for sensor networks,” *Wireless Networks*, vol. 8(5), 2002, pp. 521–534.

[34] C. Sharma and R. Vaid, “Analysis of existing protocols in WSN based on key parameters,” *In Proceedings of 2nd International Conference on Communication, Computing and Networking*, Springer, Singapore, 2019, pp. 165–171.

[35] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” *Annual Intl. Conf. on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT), Lecture Notes in Computer Science (LNCS)*, vol. 3494. Springer, Berlin, Germany, 2005, pp 457–473.

[36] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” *13th ACM Conf. on Computer and Communications Security (CCS)*, Alexandria, VA, Nov. 2006, pp 89–98.

[37] N. Attrapadung, B. Libert, and E. De Panafieu, “Expressive key-policy attribute-based encryption with constant-size ciphertexts,” *Intl. W. on Public Key Cryptography (PKC), Lecture Notes in Computer Science (LNCS)*, vol. 6571, Springer, Berlin, Germany, 2011, pp 90–108.

[38] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” *IEEE Symp. on Security and Privacy (SP)*, Berkeley, CA, May 2007, pp 321–334.

[39] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” *Intl. W. on Public Key Cryptography (PKC), Lecture Notes in Computer Science (LNCS)*, vol. 6571. Springer, Berlin, Germany, 2011, pp 53–70.

[40] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance evaluation of attribute-based encryption: Toward data privacy in the IoT," *IEEE Intl. Conf. on Communications (ICC)*, Sydney, NSW, June 2014, pp 725–730.

[41] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," *20th USENIX conf. on Security (SEC)*, San Francisco, CA, vol. 3, Aug. 2011, pp 34–34.

[42] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the Internet of Things," *Future Generation Computer Systems*, Elsevier, vol. 49, Aug. 2015, pp 104–112.

[43] L. Malina, J. Hajny, R. Fujdiak, and J. Hosek, "On perspective of security and privacy-preserving solutions in the internet of things," *Computer Networks*, Elsevier, vol. 102, June 2016, pp 83–95.

[44] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *Intl. Conf. on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT), Lecture Notes in Computer Science (LNCS)*, vol. 1592. Springer, Berlin, Germany, April 1999, pp 223–238.

[45] J. Benaloh, "Dense probabilistic encryption," *W. on Selected Areas of Cryptography*, May 1994, pp 120–128.

[46] J. Sen, "Privacy preservation technologies in Internet of Things," *Intl. Conf. on Emerging Trends in Mathematics, Technology and Management*, 2010, pp 496–504.

[47] G. Sun, G. Yang, D. Liao, Z. Xu, and K. Luo, "Efficient mapping of hybrid virtual networks across multiple domains," *IEEE Global Communications Conf. (GLOBECOM)*, San Diego, CA, Dec. 2015, pp 1–6.

[48] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen, "Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications," *IEEE Trans. on Parallel and Distributed Systems*, vol. 23(9), Sep. 2012, pp 1621–1631.

[49] S. Sicari, L. A. Grieco, G. Boggia, and A. Coen-Porisini, "DyDAP: A dynamic data aggregation scheme for privacy aware wireless sensor networks," *J. of Systems and Software*, vol. 85(1), Jan. 2012, pp 152–166.

[50] M. S. H. Talpur, M. Z. A. Bhuiyan, and G. Wang, “Shared–node IoT network architecture with ubiquitous homomorphic encryption for healthcare monitoring,” *Intl. J. of Embedded Systems*, vol. 7(1), 2014, pp 43–54.

[51] K. S. Wong and M. H. Kim, “Towards self-awareness privacy protection for Internet of Things data collection,” *J. of Applied Mathematics, Hindawi*, vol. 2014, Article ID 827959, June 2014.

[52] PCI Security Standards Council, “Initial roadmap: point-to-point encryption technology and PCI DSS compliance,” *Emerging Technology Whitepaper*, 2010. Accessed on Sep. 10, from: https://www.pcisecuritystandards.org/documents/pci_ptp_encryption.pdf

[53] L. Sweeney, “k-anonymity: A model for protecting privacy,” *Intl. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10(05), Oct. 2002, pp 557–570.

[54] J. Domingo-Ferrer and V. Torra V, “A critique of k-anonymity and some of its enhancements,” *3rd IEEE Intl. Conf. on Availability, Reliability and Security*, Barcelona, Spain, March 2008, pp 990–993.

[55] X. Huang, R. Fu, B. Chen, T. Zhang, and A. W. Roscoe, “User interactive internet of things privacy preserved access control,” *IEEE Intl. Conf. for Internet Technology and Secured Trans.*, London, England, Dec. 2012, pp 597–602.

[56] P. Gope and T. Hwang, “Untraceable sensor movement in distributed IoT infrastructure,” *IEEE Sensors J.*, vol. 15(9), Sep. 2015, pp 5340–5348.

[57] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, “Attribute-based signature and its applications,” *5th ACM Symp. on Information, Computer and Communications Security (ASIACCS)*, Beijing, China, April 2010, pp 60–69.

[58] J. Camenisch and E. Van Herreweghen, “Design and implementation of the idemix anonymous credential system,” *9th ACM Conf. on Computer and Communications Security (CCS)*, Washington, DC, Nov. 2002, pp 21–30.

[59] C. Paquin and G. Zaverucha, “U-prove cryptographic specification v1. 1. Tech. Rep.,” *Microsoft Corporation*, Dec. 2013. Accessed on Sep. 15, from: https://www.microsoft.com/enus/research/wp-content/uploads/2016/02/UProve20Cryptographic20Specification20V1.1.pdf

[60] J. Hajny and L. Malina, “Unlinkable attribute-based credentials with practical revocation on smart-cards,” *Smart Card Research and Advanced Applications (CARDIS), Lecture Notes in Computer Science (LNCS)*, vol. 7771, Springer, Berlin, Germany, 2012, pp 62–76.

[61] D. Chaum and E. Van Heyst, “Group signatures,” *W. on the Theory and Application of Cryptographic Techniques, Advances in Cryptology - EUROCRYPT, LNCS 547, Lecture Notes in Computer Science (LNCS)*, Springer-Verlag, Berlin, Germany, 1991, pp 257–265.

[62] K. El Defrawy and G. Tsudik, “Prism: Privacy-friendly routing in suspicious MANETs (and VANETs),” *IEEE Intl. Conf. on Network Protocols*, Orlando, FL, Oct. 2008, pp 258–267.

[63] A. Wasef and X. Shen, “Efficient group signature scheme supporting batch verification for securing vehicular networks,” *IEEE Intl. Conf. on Communications*, Cape Town, South Africa, May 2010, pp 1–5.

[64] Y. Zheng, “Digital signcryption or how to achieve cost (signature &encryption) cost (signature) + cost(encryption),” *In Advances in Cryptology-CRYPTO, LNCS 1294*, Heidelberg, Springer, 1997.

[65] F. Li, Z. Zheng, and C. Jin, “Secure and efficient data transmission in the Internet of Things,” *Telecommunication Systems*, Springer, vol. 62(1), May 2016, pp 111–122.

[66] F. Kargl, F. Schaub, and S. Dietzel, “Mandatory enforcement of privacy policies using trusted computing principles,” *Intelligent Information Privacy Management Symp., AAAI Spring Symp.*, Stanford University, Stanford, CA, 2010, pp 104–109.

[67] C. Dsouza, G. J. Ahn, and M. Taguinod, “Policy-driven security management for fog computing: preliminary framework and a case study,” *15th IEEE Intl. Conf. on Information Reuse and Integration (IRI)*, Redwood City, CA, Aug. 2014, pp. 16–23.

[68] N. Davies, N. Taft, M. Satyanarayanan, S. Clinch, and B. Amos, “Privacy mediators: helping IoT cross the chasm,” 17th ACM Intl. W. on Mobile Computing Systems and Applications (HotMobile), St. Augustine, FL, Feb. 2016, pp. 39–44.

[69] O. Sibert, D. Bernstein, D. Van Wie, "The DigiBox: a self-protecting container for information commerce," *In Proc. of First USENIX Workshop on Electronic Commerce*, New York, NY, 1995, pp 15–15.

[70] R. Neisse, G. Baldini, G. Steri, Y. Miyake, S. Kiyomoto, and A. R. Biswas, "An agent-based framework for informed consent in the Internet of Things," *2nd IEEE World Forum on Internet of Things (WF-IoT)*, Milan, Italy, Dec. 2015, pp 789–794.

[71] D. G. Korzun, S. I. Balandin, and A. V. Gurtov, "Deployment of smart spaces in Internet of Things: overview of the design challenges," *Internet of Things, Smart Spaces, and Next Generation Networking, Lecture Notes in Computer Science (LNCS)*, vol. 8121. Springer, Berlin, Germany, 2013, pp 48–59.

[72] A. Tchao, G. Di Marzo, J. H. Morin, "Personal DRM (PDRM)--a self-protecting content approach," *In Digital Rights Management: Technology, Standards and Applications*, F. Hartung et al., CRC Press, Taylor & Francis Group, New York, NY, 2017.

[73] P. Angin, B. Bhargava, R. Ranchal, N. Singh, M. Linderman, L. B. Othmane, and L. Lilien, "An entity-centric approach for privacy and identity management in cloud computing," *29th IEEE Symposium in Reliable Distributed Systems*, Oct. 2010, pp. 177–183.

[74] R. Salih and L. Lilien, "The active bundle scheme for protecting electronic medical records," *Transactions of the International Conference on Health Information Technology Advancement*, Vol. 1(1), Western Michigan University, Kalamazoo, MI, Oct. 2011, pp. 109–115.

[75] R. Ranchal and B. Bhargava, "Protecting PLM data throughout their lifecycle," *In International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, Springer, Berlin, Heidelberg, Jan. 2013, pp. 633–642.

[76] F. Hosseinpour, J. Plosila, and H. Tenhunen, "An approach for smart management of big data in the fog computing context," *In IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec. 2016, pp. 468–471.

[77] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience*, 47(9), 2017, pp.1275-1296.

[78] B. Dickson, “How fog computing pushes IoT intelligence to the edge,” *Crunch Network*, 2016. Accessed on June 9, 2017, from: https://techcrunch.com/2016/08/02/how-fog-computing-pushes-iot-intelligence-to-the-edge/

[79] E. M. Tordera, X. Masip-Bruin, J. Garcia-Alminana, A. Jukan, G. J. Ren, J. Zhu, and J. Farre, “What is a fog node? A tutorial on current concepts towards a common definition,” *arXiv preprint*, vol. arXiv:1611.09193v1, Nov. 2016.

[80] P. A. Hsiung, “Real-Time Constraints,” 2001. Accessed on July 28, 2018, from: https://www.cs.ccu.edu.tw/~pahsiung/publications/papers/rtc.pdf

[81] A. Al-Hasnawi and L. Lilien, “Pushing data privacy control to the edge in IoT using policy enforcement fog module,” *3rd Intl. Symp. on Real-Time Data Processing for Cloud Computing (RTDPCC II)*, in conjunction with the *10th IEEE/ACM Intl. Conf. on Utility and Cloud Computing (UCC)* Austin, TX, Dec. 2017, pp. 145–150.

[82] A. Anderson et al., “Extensible access control markup language (XACML) version 1.0,” *Organization for the Advancement of Structured Information Standards (OASIS)*, Burlington, MA, Feb. 2003.

[83] D. Ferraiolo, R. Chandramouli, R. Kuhn, and V. Hu, “Extensible access control markup language (XACML) and next generation access control (NGAC),” *ACM Intl. W. on Attribute-Based Access Control (ABAC)*, New Orleans, LA, March 2016, pp. 13–24.

[84] “Foscam home security.” Accessed on July 28, 2018 from: https://www.foscam.com/

[85] S. W. Ibrahim, “A comprehensive review on intelligent surveillance systems,” *Communications in Science and Technology*, 1(1), May 2016.

[86] J. Carlsen, “Best home surveillance systems of 2018.” Accessed on Sep. 28, 2018 from: https://www.toptenreviews.com/home/smart-home/best-home-surveillance-systems/

[87] J. Black, T. Ellis, and P. Rosin, “Multi view image surveillance and tracking,” *IEEE Workshop on Motion and Video Computing*, Dec. 2002, pp. 169–174.

[88] “Foscam home security- FOSCAM E1.” Accessed on Sep. 28, 2018 from: https://www.foscam.com/E1.html

[89] A. Jacobsson, M. Boldt, and B. Carlsson, “A risk analysis of a smart home automation system,” *Future Generation Computer Systems*, 56, 2016, pp. 719–733.

[90] D. Christin, A. Reinhardt, S. S, Kanhere, and M. Hollick, “A survey on privacy in mobile participatory sensing applications,” *Journal of systems and software*, 84(11), 2011, pp. 1928–1946.

[91] M. J. O'Mahony, S. Dimitra, K. H. David, and T. Anna, “The application of optical packet switching in future communication networks,” *IEEE Communications magazine 39*, no. 3, 2001, pp. 128–135.

[92] A. Khanna and T. Ravi, “IoT based interactive shopping ecosystem,” *In 2nd IEEE International Conference on Next Generation Computing Technologies (NGCT)*, 2016, pp. 40–45.

[93] M. Taneja and D. Alan, “Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm,” *In IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 1222–1228.

[94] “Instructions per second,” *Wikipedia*. Accessed on May 6, 2018, from: https://en.wikipedia.org/wiki/Instructions_per_second.

[95] J. Kurose and K. Ross, “Computer Networking: A Top-Down Approach,” 7th edition, Pearson, Boston, MA, 2016.

[96] S. T. Mirtchev, “Packet-level link capacity evaluation for IP networks,” *Cybernetics and Information Technologies*, 18(1), 2018, pp.30–40.

[97] “Statista.” Accessed on Sep. 28, 2018 from: https://www.statista.com/statistics/183648/average-size-of-households-in-the-us/

[98] J. Raj, “Art of computer systems performance analysis techniques for experimental design measurements simulation and modeling,” *Wiley Computer Publishing*, John Wiley & Sons, Inc., 1991.

[99] “Poisson distribution.” Accessed on Sep. 19, 2018, from: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.poisson.html/numpy.random.poisson

[100] “Discrete uniform distribution.” Accessed on Sep. 19, 2018, from: https://docs.scipy.org/doc/scipy/reference/tutorial/stats/discrete_randint.html

[101] “Binomial distribution.” Accessed on Sep. 19, 2018, from: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.binomial.html#numpy.random.binomial