Towards a new data privacy-based approach for IoT

Faiza Loukil

N°d'ordre NNT : 2019LYSE3044

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
L'Université Jean Moulin Lyon 3

Ecole Doctorale N° 512
ED en Informatique et Mathématiques (InfoMaths)

Discipline de doctorat : Informatique

Soutenue publiquement à 14h le 15/10/2019, par :

Faiza LOUKIL

Towards a new data privacy-based approach for IoT

Devant le jury composé de :

Harald KOSCH, Professeur des universités, Université Passau-Allemagne	Rapporteur
Michael MRISSA, Professeur des universités, InnoRenew CoE, Université de Primorska, Slovénie	Rapporteur
Benjamin NGUYEN, Professeur des universités, INSA Centre Val de Loire, Bourges	Examinateur
Genoveva VARGAS SOLAR, Chargée de Recherche HDR, CNRS, Laboratoire LIG	Examinatrice
Chirine GHEDIRA-GUEGAN, Professeur des universités, IAELyon Université Lyon3	Directrice de thèse
Aïcha-Nabila BENHARKAT, Maître de Conférences, INSA de Lyon	Co-directrice de thèse

Acknowledgments

At the beginning of this dissertation and the end of this journey, I would like to thank all these who helped me to complete this work. Firstly, I would like to thank my academic supervisor, Mrs. Chirine GHEDIRA-GUEGAN for the numerous scientific discussions, the insightful inputs on my work, and her guidance all along my thesis. I would also like to express my gratitude to Mrs. Aïcha-Nabila BENHARKAT for her interesting comments, advices, and reflections. Both have provided me with valuable academic suggestions, relevant research ideas, and administrative supports during my research.

I would also like to thank the members of the jury who evaluated my work: Mr. Harald KOSCH and Mr. Michael MRISSA for reviewing my dissertation, as well as Mrs. Genoveva VARGAS SOLAR and Mr. Benjamin NGUYEN for being part of my jury and their interest in my work.

I would like to express my gratitude to Mrs. Khouloud BOUKADI for her appreciate collaboration in my research publications in academic conferences over the past three years and her encouragements when they were the most needed.

I would also like to thank my fellow PhD students in SOC team for all the passionate discussions, the not scientific support, and the shared trips.

Finally, I thank my lovely family in Sfax (Tunisia) for their constant support. Thanks to their encouragement, comprehension, and support, I keep moving forward to achieve my objective and complete this amazing journey.

Abstract

The Internet of Things (IoT) connects and shares data collected from smart devices in several domains, such as smart home, smart grid, and healthcare. According to Cisco, the number of connected devices is expected to reach 500 Billion by 2030. Five hundred zettabytes of data will be produced by tremendous machines and devices. Usually, these collected data are very sensitive and include metadata, such as location, time, and context. Their analysis allows the collector to deduce personal habits, behaviors and preferences of individuals. Besides, these collected data require the collaboration of several parties to be analyzed. Thus, due to the high level of IoT data sensitivity and lack of trust on the involved parties in the IoT environment, the collected data by different IoT devices should not be shared with each other, without enforcing data owner privacy. In fact, IoT data privacy has become a severe challenge nowadays, especially with the increasing legislation pressure.

Our research focused on three complementary issues, mainly (i) the definition of a semantic layer designing the privacy requirements in the IoT domain, (ii) the IoT device monitoring and the enforcement of a privacy policy that matches both the data owner's privacy preferences and the data consumer's terms of service, and (iii) the establishment of an end-to-end privacy-preserving solution for IoT data in a decentralized architecture while eliminating the need to trust any involved IoT parties.

To address these issues, our work contributes to three axes. First, we proposed a new European Legal compliant ontology for supporting preserving IoT PrivacY, called LIoPY that describes the IoT environment and the privacy requirements defined by privacy legislation and standards. Then, we defined a reasoning process whose goal is generating a privacy policy by matching between the data owner's privacy preferences and the data consumer's terms of service. This privacy policy specifies how the data will be handled once shared with a specific data consumer. In order to ensure this privacy policy enforcement, we introduced an IoT data privacy-preserving framework, called PrivBlockchain, in the second research axis. PrivBlockchain is an end-to-end privacy-preserving framework that involves several parties in the IoT environment for preserving IoT data privacy during the phases of collection, transmission, storage, and processing. The proposed framework relied on, on the one hand, the blockchain technology, thus supporting a decentralized architecture while eliminating the need to trust any involved IoT parties and, on the other hand, the smart contracts, thus supporting a machine-readable and self-enforcing privacy policy whose goal is to preserve the privacy during the whole data life-cycle, covering the collection, transmission, storage and processing phases. Finally, in the third axis, we designed and implemented the proposal in order to prove its feasibility and analyze its performances.

Key-Words: privacy, Internet of Things (IoT), ontology and inference, blockchain technology.

Résumé

Les objets connectés collectent et partagent des données dans différents domaines tels que les maisons intelligentes, les réseaux de distribution d'électricité intelligents et la santé. Selon Cisco, le nombre d'objets connectés devrait atteindre 500 milliards d'ici 2030 avec une quantité de données produites d'environ cinq cents zettaoctets. Toutefois, ces données recueillies sont généralement très riches et comprennent souvent des métadonnées telles que l'emplacement, l'information temporelle et le contexte, rendant ainsi possible de déduire facilement les habitudes personnelles, les comportements et les préférences des individus. De plus, l'analyse de ces données recueillies nécessite la collaboration de plusieurs intervenants. Ainsi, en raison du niveau élevé de la sensibilité des données et du manque de confiance entre les parties impliquées dans un tel réseau, ces données ne doivent pas être partagées, sans que la vie privée du propriétaire des données soit respectée. En effet, la protection de la vie privée des données issues des objets connectés est devenue un défi majeur, en particulier avec la pression croissante de la législation.

Nos travaux de recherche sont focalisés sur trois problématiques complémentaires qui sont la problématique de la modélisation des exigences de la protection de la vie privée, la problématique de monitoring les objets connectés et la garantie du respect d'une politique commune qui correspond à la fois aux préférences des propriétaires des données et aux conditions des consommateurs des données, et enfin la problématique de protection de la vie privée durant tout le cycle de vie des données générées par ces objets dans une architecture décentralisée qui élimine le besoin de faire confiance aux parties impliquées dans le réseau d'objets connectés.

Afin de répondre à ces problématiques, nous avons proposé dans un premier lieu une ontologie appelée LIoPY qui modélise la métadonnée ainsi que les contraintes de manipulation des données en adéquation avec les normes et les lois de protection de la vie privée. Puis, pour aligner sémantiquement les exigences en matière de protection de la vie privée des propriétaires des données ainsi que des consommateurs des données, nous avons étendu l'ontologie par des relations sémantiques d'arborescence et des règles sémantiques d'inférence qui génèrent une politique de protection de la vie privée commune. Cette politique décrit comment les données doivent être manipulées une fois partagées avec un consommateur donné. Afin de garantir le respect de cette politique commune, nous avons introduit le framework PrivBlockchain, un framework qui implique toutes les parties intervenantes dans un réseau d'objets connectés dans la protection des données qui en sont issues lors des phases de collecte, du transfert, du stockage jusqu'à la phase de l'utilisation ou bien l'analyse. Le framework proposé repose, d'une part, sur la technologie de la blockchain d'où le support d'une architecture décentralisée, tout en éliminant le besoin de faire confiance aux parties impliquées dans le réseau d'objets connectés et, d'autre part, sur les contrats dits « intelligents » d'où le support d'une politique auto-appliquée et lisible par la machine. Son rôle est de protéger la vie privée lors des phases de collecte, du transfert, du stockage jusqu'à la phase de l'analyse des données issues des objets connectés. Enfin, nous avons validé notre proposition par l'élaboration et l'implantation d'un prototype afin de prouver sa faisabilité et analyser ses performances.

Mots clés : protection de la vie privée, objets connectés, ontologie et inférence, technologie de la blockchain.

Contents

Abstract v

Résumé vii

List of figures xiii

List of tables xv

List of algorithms xvii

Introduction 1
1 Context . . . 1
2 Research Problem statement . . . 3
3 Contributions summary . . . 5
4 Thesis approach . . . 7
5 Organization of the dissertation . . . 8

Chapter 1 Related Work 9
1.1 Introduction . . . 9
1.2 Privacy . . . 10
1.2.1 Privacy legislation . . . 11
1.2.2 Privacy design strategies . . . 13
1.2.3 Three-layered privacy model . . . 15
1.2.4 Privacy-preserving architectures . . . 15
1.2.5 Privacy-preserving mechanisms . . . 16
1.2.6 Related privacy-preserving approaches in several domains . . . 19
1.3 Privacy-preserving approaches in the IoT domain . . . 22
1.3.1 Centralized-based approaches . . . 22
1.3.2 Distributed-based approaches . . . 24
1.3.3 Trusted Third-Party-based approaches . . . 28
1.4 Analysis . . . 30
1.5 Summary . . . 33

Chapter 2 LIoPY: European Legal compliant ontology for supporting preserving IoT PrivacY 35
2.1 Introduction . . . 35
2.2 Existing IoT and privacy ontologies . . . 36
2.3 Overview of the European Legal compliant ontology for supporting preserving IoT PrivacY (LIoPY) . . . 41
2.3.1 Building LIoPY . . . 41
2.3.2 LIoPY's Modules . . . 50
2.4 Privacy Preferences Model . . . 52

2.4.1 Collected-data-centric privacy-preserving perspective . . . 53
2.4.2 Shared-data-centric privacy-preserving perspective . . . 54
2.5 Summary . . . 69

Chapter 3 Semantic Rule Manager: Reasoning process validation 71
3.1 Introduction . . . 71
3.2 Reasoning process . . . 72
3.2.1 Privacy Attribute Matching algorithm definition . . . 72
3.2.2 Privacy Attribute Matching algorithm implementation . . . 75
3.2.3 Privacy Attribute Matching algorithm validation: Semantic Rule Manager . . . 79
3.3 Experimentation . . . 81
3.3.1 Motivating scenario . . . 81
3.3.2 Experimental environment . . . 81
3.3.3 Experimental results . . . 82
3.4 Summary . . . 86

Chapter 4 PrivBlockchain: Blockchain-based IoT data privacy-preserving framework 87
4.1 Introduction . . . 88
4.2 Background and Related Work . . . 88
4.2.1 Blockchain technology . . . 88
4.2.2 Homomorphic encryption technology . . . 90
4.3 Design goals . . . 91
4.4 PrivBlockchain: IoT data privacy-preserving framework . . . 92
4.4.1 PrivBlockchain overview . . . 92
4.4.2 PrivBlockchain core components . . . 95
4.4.3 PrivBlockchain modules . . . 99
4.5 Blockchain-based IoT device management module . . . 100
4.5.1 Smart contract description . . . 100
4.5.2 Privacy permission setting adding process . . . 101
4.6 Blockchain-based IoT data sharing module . . . 107
4.6.1 Smart contract description . . . 108
4.6.2 IoT data sharing process . . . 108
4.7 Homomorphic encryption-based IoT data aggregation module . . . 113
4.7.1 Smart contract description . . . 114
4.7.2 Privacy policy generation process . . . 114
4.7.3 IoT data aggregation process . . . 116
4.8 Summary . . . 120

Chapter 5 Evaluation and Analysis 121
5.1 Introduction . . . 121
5.2 Evaluation . . . 122
5.2.1 Motivating scenarios . . . 122
5.2.2 Experimental environment . . . 123
5.3 Blockchain-based smart home . . . 123
5.3.1 IoT device management use case . . . 123
5.3.2 Security and privacy analysis . . . 127
5.3.3 Performance Evaluation . . . 128
5.4 Blockchain-based healthcare system . . . 130
5.4.1 IoT data sharing use case . . . 130
5.4.2 Security and privacy analysis . . . 134
5.4.3 Performance Evaluation . . . 135
5.5 Blockchain-based smart grid . . . 138
5.5.1 IoT data aggregation use case . . . 139
5.5.2 Security and privacy analysis . . . 140
5.5.3 Performance Evaluation . . . 141
5.6 Comparative performance analysis . . . 144

5.7 Summary: privacy design strategies compliance 149

Conclusion and Future Work **151**
1 Context and objectives 151
2 Proposed contributions 152
3 Limitations of this work 153
4 Future Work 154

Bibliography **162**

Appendix A LIoPY's evaluation using OntoMetrics platform **163**

Appendix B Smart contract-based misbehavior detection results **167**

List of Figures

1 Thesis Approach 7

2.1 An overview of SOUPA [Chen et al., 2004] 36
2.2 An overview of SSN ontology [Haller et al., 2018] 37
2.3 An overview of IoT-Lite ontology [Bermudez-Edo et al., 2017] 38
2.4 An overview of PPO [Sacco and Passant, 2011] 38
2.5 An overview of DPO [Bawany and Shaikh, 2017] 39
2.6 An overview of ORDM [Wang et al., 2016] 40
2.7 Informal text analysis of the GDPR [GDPR, 2016] 43
2.8 LIoPY's conceptual model document: concepts and relationships 45
2.9 LIoPY's Modules 50
2.10 IoT Description Module 51
2.11 a) IoT Resource Management Module, b) IoT Resource Result Sharing Management Module 52
2.12 Example of a privacy permission setting instance 54
2.13 LIoPY's principal classes 55
2.14 Example of the Consent class instantiation 56
2.15 Example of a purpose tree 57
2.16 Example of a retention individual 58
2.17 Example of an operation tree 59
2.18 Condition class: Ancestor and Descendants 60
2.19 Example of a disclosure tree 62
2.20 Example of a data category tree 63
2.21 Example of a privacy rule instance 65
2.22 Example of terms of service instance 67
2.23 Example of a privacy policy instance 68

3.1 Example of a privacy matching SWRL rule for instantiating a Privacy_Policy . . . 77
3.2 Architecture of Semantic Rule Manager 80
3.3 Privacy policy sharing process 80
3.4 After privacy policy derivation 83
3.5 Processing time of privacy policy derivation 84
3.6 CPU usage 85
3.7 Used memory size 85

4.1 PrivBlockchain Architecture 94
4.2 Architecture of Semantic IoT Gateway 99
4.3 Add new IoT device business process notated in BPMN 102
4.4 Transaction creation sub-process notated in BPMN 103
4.5 Private ledger mining sub-process notated in BPMN 104
4.6 Smart contract generation sub-process notated in BPMN 104
4.7 Smart contract function invocation sub-process notated in BPMN 105

4.8 IoT data sharing business process notated in BPMN . . . 110
4.9 Blockchain mining sub-process notated in BPMN . . . 111
4.10 Terms of service treatment sub-process notated in BPMN . . . 112
4.11 File creation sub-process notated in BPMN . . . 112
4.12 GrantPermission transaction creation sub-process notated in BPMN . . . 113
4.13 SendHash transaction creation sub-process notated in BPMN . . . 113
4.14 Privacy policy generation process . . . 115
4.15 IoT data aggregation process . . . 116

5.1 Blockchain-based smart home test system screenshot during privacy permission setting definition . . . 124
5.2 Blockchain-based smart home test system screenshot during privacy permission setting verification . . . 125
5.3 Blockchain-based smart home test system screenshot in case of denial of service from one blockchain address . . . 126
5.4 Blockchain-based smart home test system screenshot in case of denial of service from several blockchain addresses . . . 126
5.5 Computational cost of privacy permission setting definition and verification for one smart device . . . 128
5.6 Average computational cost of privacy permission setting definition and verification for five smart devices . . . 129
5.7 Average computational cost of privacy permission setting definition and verification for fifty smart devices . . . 129
5.8 Blockchain-based healthcare test system screenshot during registration time . . . 131
5.9 Blockchain-based healthcare test system screenshot in case of an emergency . . . 132
5.10 Blockchain-based healthcare test system screenshot in case of privacy violation attempts . . . 133
5.11 Average processing time with different invoked functions . . . 136
5.12 Average processing time of **addFile** function with different file size . . . 136
5.13 Average processing time of **updateFile** function with different file size . . . 137
5.14 Average *gas* usage with different file size . . . 138
5.15 Blockchain-based smart grid test system screenshot . . . 139
5.16 Computational cost of data aggregation and data decryption . . . 142
5.17 Data aggregation's computational cost details . . . 143
5.18 Communication overhead . . . 144

B.1 Results after misbehavior occurring twice by [Zhang et al., 2018] system. . . . 167
B.2 Results after misbehavior occurring twice by the proposed smart home system. . . 168

List of Tables

1.1 Mapping of privacy design strategies onto privacy legal principles 14
1.2 List of the privacy-preserving approaches in several domains 21
1.3 List of the privacy-preserving approaches in the IoT domain 31
1.4 List of the privacy-preserving approaches in the IoT domain (continued) 32

2.1 Existing IoT ontologies comparison 40
2.2 LIoPY's specification document 42
2.3 LIoPY's knowledge acquisition document 44
2.4 LIoPY's conceptual model document: data property dictionary 46
2.5 LIoPY's integration document 47
2.6 LIoPY's evaluation document using Quality Metrics and Quality Criteria Matrix . 49

3.1 Notations 72

5.1 Cost Overhead 138
5.2 Computation complexity of IoT data aggregation test system 142
5.3 Comparative study between aggregation-based approaches and the proposed smart grid test system 147
5.4 Used variables in the comparison 148
5.5 Comparison of eavesdropping probability of different system components 148
5.6 List of the three proposed privacy-preserving test systems 150

List of Algorithms

1 Privacy policy generation through privacy requirement matching. 74

2 IoT device misbehavior judge. 105
3 IoT device misbehavior detection. 106

Introduction

Table of Contents

1 **Context** 1
2 **Research Problem statement** 3
3 **Contributions summary** 5
4 **Thesis approach** 7
5 **Organization of the dissertation** 8

1 Context

The Internet of Things (IoT) has emerged as one of the most significant technology in recent years. Its wide deployment has improved the quality of the individual's lifestyle by providing better facilities on various daily applications, such as smart home, smart grid, and smart city. The IoT's benefit to individuals' lives is realized thanks to the data analytics from the smart devices and the huge volumes of produced IoT data. From the point of view of the private user [1], the most obvious effects of the IoT introduction will be visible in both working and domestic fields, such as assisted living, e-health, and enhanced learning. Similarly, from the perspective of the business users [2], the most apparent consequences will be equally visible in fields, such as automation and industrial manufacturing, business/process management, and intelligent transportation of people and goods [Atzori et al., 2010].

According to Cisco [Cisco, 2016], 500 billion devices are expected to be connected to the Internet by 2030. Five hundred zettabytes of data will be produced by tremendous machines and devices. Actually, many challenging issues related to the IoT device characteristics still need to be addressed. In fact, they have a low computation and an energy capacity, so they can easily be attacked when connected to the Internet. Indeed, security and privacy techniques, such as encryption, authentication, and role-based access control which are used in the context of conventional information systems proved to be very expensive when running on devices with limited computing capabilities in the IoT domain [Singla et al., 2015]. In order to overcome this

[1] interchangeably termed data owner in this thesis.
[2] interchangeably termed data consumer in this thesis.

problem, another IoT device type is proposed, called IoT gateway. Its role is to collect and send the produced data from IoT sensors and actuators to the data consumers' external platforms to be remotely analyzed. However, the users desire a more adapted IoT gateway that can improve the IoT data privacy preservation before sending them to these external platforms. Thus, an IoT gateway that enables a better control over the set of private IoT resources and protects the collected personal data and their privacy is required.

In the IoT domain, multiple devices collect, exchange, store, and process large amount of fine-granularity and high-frequency data in every aspect of life. Such detailed data improve delivering advanced services in a wide range of application domains. However, the produced IoT data are generally rich in sensitive data and their analysis allows the data consumer to deduce data owner-personal behaviors, habits and preferences. For instance, a smart meter is used to monitor electricity usage and transmit these data back to the utility. However, if compromised, the meter can be exploited by criminals who analyze the produced data to determine when owners are away, thus finding the best targets and times for a break-in. Thus, IoT raises concerns about privacy and data protection. Indeed, collecting data in IoT applications increases the data owner's worries about the potential uses of these data. In fact, the data owner wish not share the produced IoT data with other competitor organizations without retaining some level of control.

Due to the resource-constrained IoT devices and the high sensitive IoT data, security and privacy concerns rise in the IoT domain. According to [Clarke, 2006], the privacy of personal information involves the right to control when, where, how, to whom, and to what extent an individual shares the own personal information, as well as the right to access personal information given to others, to correct them. All these privacy requirements exist in the European regulation [GDPR, 2016] and privacy standards [ISO/IEC29100, 2011][OECD, 1981]. In this thesis, 'privacy requirements' means the obligations that must be fulfilled by all the involved parties, including the data owner and the data consumer in the process of personal data treatment to preserve the privacy during the whole data lifecycle, covering the collection, transmission, storage and processing phases.

Despite the increasing legislation pressure, the data owners have a little or no control over the collected data about themselves [Maddox, 2015]. For instance, sharing the collected data by wearable devices with service providers leads to lose the IoT data control and ownership [Maddox, 2015]. Moreover, these collected data require the collaboration of several parties to be analyzed. However, due to the high level of IoT data sensitivity and lack of trust on the involved parties in the IoT environment, the collected data by different IoT devices should not be shared with each other, without enforcing the data owner's privacy. Moreover, the used IoT gateways are generic, with basic settings, and do not preempt the user's requirements especially concerning the privacy issue.

The remainder of this chapter explores the thesis from multiple dimensions, including the research questions, the contributions, and the dissertation organization.

2 Research Problem statement

In this context, the research problem that we address in this thesis is:

How to preserve IoT data privacy while considering both the whole process of collecting, transmitting, storing, and processing IoT data as well as the compliance with existing privacy laws and data protection regulation?

We decompose this research problem into four sub-problems.

RP1 - Evaluating and comparing the privacy-preserving approaches in the IoT domain.

In the European Union, the legal right to privacy is based on Article 8 of the European Convention of Human Rights of 1950 [Pearson, 2009]. In the context of data protection, this right has been made explicit in the data protection directive of 1995 [Directive 95/46/EC, 1995] and then enhanced by the general data protection regulation in 2016 [GDPR, 2016]. However, there is a considerable gap between privacy legal requirements and engineering. Indeed, not every legal requirement can be met by a system design pattern. For instance, the legal requirement, called "Legitimacy of processing" is a good example. If the processing is illegitimate, then it will be illegitimate irrespective of the design of the system [Hoepman, 2014]. Thus, only the legal requirements that can be implemented are used as privacy analysis criteria to evaluate the existing privacy-preserving approaches. Besides privacy legislation, other privacy analysis criteria can be extracted based on the review of literature where each work proposed its own evaluation criteria.
Therefore, we consider the following research questions:

- Which privacy analysis criteria to use to evaluate and compare the privacy-preserving approaches?
- Which are the neglected areas by the existing privacy-preserving approaches that need more investigation?

RP2 - Designing privacy requirements in the IoT domain.

The privacy violation risks and the legislation pressure push both the data owner and the data consumer to preserve the privacy of the shared data. However, matching both the data owner's preferences and the data consumer's terms of service requires the use of the same privacy vocabulary that describes the privacy requirements. This matching enables the creation of a common privacy policy that can be applied to preserve the data owner privacy in the IoT environment while handling the shared data.
Therefore, we consider the following research questions:

- How to describe the privacy requirements using the same terms by both the data owner and the data consumer?

- How to match between the data owners' privacy preferences and the data consumers' terms of service?

RP3 - Defining self-enforcing privacy requirements in the IoT domain.

Actually, many challenging issues related to the IoT device characteristics still need to be addressed. In fact, they have low computation and energy capacity to protect personal data and the user's privacy. One existing solution is to manage the IoT devices using a device with high computation capacity, called IoT gateway whose role is to collect and send the produced data from IoT sensors and actuators to the data consumers to be remotely analyzed. However, the used IoT gateways are generic, with basic settings, and do not preempt the data owner's requirements especially concerning the privacy issue.

The data consumers' platforms gather the IoT data and use them to personalize services, optimize decision-making processes, and predict future trends. However, the data owners have no guarantee that the data consumer will respect the licensing agreement concerning privacy and security protection [Maddox, 2015].

In order to prevent the data consumer from learning individual data, several data owners can collaborate among them, aggregate their produced IoT data, and only send the obtained result. However, several challenges remain to be addressed to takle the privacy issue during the data aggregation process. Indeed, eliminating the single point of failure, the single point of trust, and the raw data disclosure are the main problems that are needed to be eliminated in order to offer an end-to-end privacy-preserving solution in the IoT domain.
Therefore, we consider the following research questions:

- How to enforce the data owner's control over the owned smart devices?
- How to transform the privacy requirements and obligations to a common privacy policy that is immutable and self-enforcing?
- How to raise the individual's privacy during the whole IoT data lifecycle while eliminating the raw data disclosure issue?

RP4 - Experimenting with privacy-preserving test systems.

While creating a test system to validate the proposed contributions that aim at addressing the third research problem (RP3), potential issues can emerge during the implementation phase. Moreover, analyzing the proposal's performance demonstrates whether the solution can be used in practice or it is cost-expensive, so cannot be used in practice.
Therefore, we consider the following research questions:

- Is the proposed model implementable?
- What performance dœs each test system provide, in terms of processing time, scalability, and cost per transaction?

- Dœs the implementation demonstrate a cost-effective approach that can be used in practice?

3 Contributions summary

This dissertation includes four contributions, each one addresses one of the defined research sub-problems.

C1 - A state-of-the-art of the privacy-preserving approaches in the IoT domain. In response to RP1.

Based on privacy guidelines, data protection laws, privacy framework, and privacy-related research, we introduce some privacy analysis criteria used to compare the privacy-preserving approaches in the literature. These criteria include privacy design strategies, a three-layered privacy model, privacy-preserving architectures, and privacy-preserving mechanisms. Moreover, we survey the main existing privacy-preserving approaches in different domains and we compare them. After that, we provide a review of the literature on privacy-preserving approaches proposed in the IoT domain, and we analyze them using our defined privacy analysis criteria. This analysis shows the lack of an end-to-end solution for privacy in the IoT domain that is compliant with the privacy legislation.

This contribution is the basis for the following publication in the international conference on Cooperative Information Systems (CoopIS 2017):

- Faiza Loukil, Chirine Ghedira-Guegan, Aïcha Nabila Benharkat, Khouloud Boukadi, and Zakaria Maamar. Privacy-aware in the IoT applications: A systematic literature review. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", pages 552–569. Springer, 2017. [Loukil et al., 2017]

C2 - A semantic system to assist in generating privacy policy. In response to RP2.

We propose LIoPY, a european Legal compliant ontology for supporting preserving IoT PrivacY that describes the IoT environment and the privacy requirements. To achieve this, LIoPY imports some concepts from standard ontologies and extends them with new concepts based on privacy legislation. To guarantee a well-built privacy ontology, we follow the MethOntology methodology [Fernández-López et al., 1997] during LIoPY's process of building. Moreover, we define a reasoning process whose goal is to match between the data owners' privacy preferences and the data consumers' terms of service. This matching enables the creation of a common privacy policy that can be applied to preserve the data owner privacy in the IoT environment while handling the shared data. We implement both LIoPY ontology and the reasoning process, evaluate LIoPY's quality, and analyze the reasoning process's performance.

This contribution is the basis for the following publication in the IEEE International Workshop on Security Aspects in Processes and Services Engineering (SAPSE 2018) proposed by the IEEE International Conference on Computers, Software and Applications (COMPSAC 2018):

- Faiza Loukil, Chirine Ghedira-Guegan, Khouloud Boukadi, and Aïcha Nabila Benharkat. Liopy: A legal compliant ontology to preserve privacy for the internet of things. In 2018 10th IEEE International Workshop on Security Aspects in Processes and Services Engineering (SAPSE), IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), pages 701–706. IEEE, 2018. [Loukil et al., 2018a]

C3 - A distributed system to assist in preserving privacy from an end-to-end view. In response to RP3.

We introduce PrivBlockchain, an end-to-end privacy-preserving framework for IoT data based that aims at addressing (i) user's control enforcement over the owned smart devices, (ii) privacy requirements and obligation compliance between untrusted parties in the IoT environment, and (iii) individual's privacy rise using group-level IoT data aggregation. To achieve this, PrivBlockchain relies on semantic, blockchain, and homomorphic encryption technologies. Thus, it includes three modules, each one addresses one of the aforementioned goals by defining a set of smart contracts. Moreover, we detail the core processes of each PrivBlockchain's module.

This contribution is the basis for the following publications in the international conference on Web Information Systems Engineering (WISE 2018) and the international conference on Cooperative Information Systems (CoopIS 2018):

- Faiza Loukil, Chirine Ghedira-Guegan, Khouloud Boukadi, and Aïcha Nabila Benharkat. Towards an end-to-end IoT data privacy-preserving framework using blockchain technology. In International Conference on Web Information Systems Engineering, pages 68–78. Springer, 2018. [Loukil et al., 2018c]

- Faiza Loukil, Chirine Ghedira-Guegan, Khouloud Boukadi, and Aïcha Nabila Benharkat. Semantic IoT gateway: Towards automated generation of privacy-preserving smart contracts in the internet of things. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", pages 207–225. Springer, 2018. [Loukil et al., 2018b]

C4 - An evaluation through test systems. In response to RP4.

We implement the defined smart contracts and validate them in a blockchain test network. Our evaluation aims at (i) proving that the proposed PrivBlockchain's model is implementable, (ii) providing a performance analysis in terms of processing time, scalability, and cost per transaction, and (iii) deciding whether our solution can be used in practice or it is cost-expensive. To achieve this, we implement our smart contracts using the Solidity language, deploy them to the Ethereum test network, and create three test systems using Truffle development framework. We show that our PrivBlockchain can handle a large variety of scenarios in different IoT domains.

This contribution is the basis for the following submission in the IEEE Transactions on Services Computing journal:

- Faiza Loukil, Chirine Ghedira-Guegan, Aïcha Nabila Benharkat, and Khouloud Boukadi. Privacy-Preserving IoT Data Aggregation Based on Blockchain and Homomorphic Encryption. In 2019 IEEE Transactions on Services Computing.

4 Thesis approach

The aim of this thesis is to propose an original approach that consists in proposing an IoT data privacy-preserving solution taking into consideration (i) the whole process of collecting, transmitting, storing and processing IoT data, (ii) the matching between the data owner privacy preferences and the data consumer queries by generating a common privacy policy for each shared data, and (iii) the enforcement of the privacy requirements and obligations in a decentralized architecture while eliminating the need to trust any involved IoT parties.
Figure 1 depicts the overall approach of this thesis. The systematic literature review was published in [Loukil et al., 2017] and is not included in this dissertation. Nevertheless, the obtained results are used as part of the literature review of the existing privacy-preserving approaches in the IoT domain (labeled *Structure Survey*). The outcome of this step is used to identify a set of challenges to be addressed in the rest of this work. Thus, we build a new ontology, called LIoPY, define a reasoning process, and prototype this proposed part. Based on the analysis of the semantic part, we define a PrivBlockchain framework, create test systems, and integrate the first prototype. Based on the tests of the second prototype, a set of experimental results are created and analyzed before summarizing this thesis.

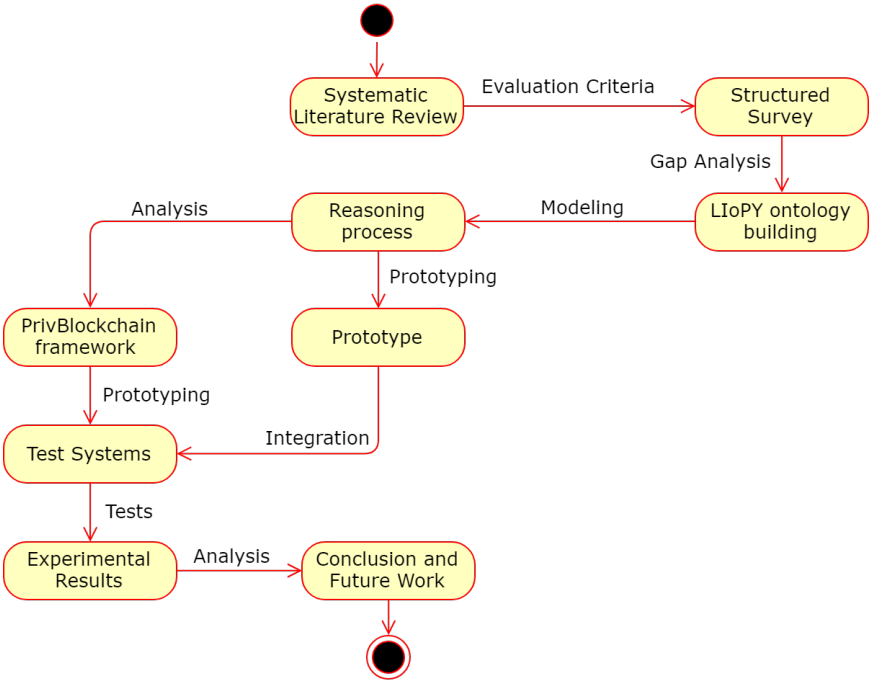


Figure 1: Thesis Approach

5 Organization of the dissertation

The remaining of this dissertation is organized on five chapters as follows. Chapter 1 gives an overview of the privacy notion in the IoT domain and analyzes the state-of-the-art of privacy-preserving approaches in the IoT domain. Chapters 2, 3, 4, and 5 detail our proposed contributions.
Below, we give more details about the aspects revolved by each chapter:

- **Chapter 1 - Related Work**: is a thorough view of privacy in different domains with focus on the IoT domain. First, we introduce a set of privacy analysis criteria that will be used to compare the existing approaches. Second, we survey the privacy-preserving approaches proposed in the IoT domain. Finally, we analyze the presented state-of-the-art based on the privacy analysis criteria defined above. We motivate our contributions from outlining strengths and weaknesses of the studied review of literature.

- **Chapter 2 - LIoPY: European Legal compliant ontology for supporting preserving IoT PrivacY**: introduces LIoPY, a european Legal compliant ontology for supporting preserving IoT PrivacY that provides a solution for overcoming the existing ontology-based model limitations. Therefore, we build the LIoPY ontology by following the MethOntology methodology [Fernández-López et al., 1997] that includes seven activities, including specification, knowledge acquisition, conceptualization, integration, implementation, evaluation, and documentation.

- **Chapter 3 - Semantic Rule Manager: Reasoning process validation**: details the implementation of our reasoning process based on the defined LIoPY ontology. The reasoning process is designed with the goal of achieving a better matching between the data owners' privacy preferences and the data consumers' terms of service. We study an experimental evaluation by analyzing the reasoning process's performance.

- **Chapter 4 - PrivBlockchain: Blockchain-based IoT data privacy-preserving framework**: unveils PrivBlockchain, an IoT data privacy-preserving framework that includes three modules based on semantic, blockchain, and homomorphic encryption technologies. PrivBlockchain aims at addressing a set of design goals in order to preserve the user's privacy in an IoT environment.

- **Chapter 5 - Evaluation and Analysis**: details the implementation and the validation of the PrivBlockchain's core components. We show that the PrivBlockchain framework is implementable while providing security analysis and performance evaluation.

The conclusion and the endeavors summarize this dissertation by presenting an assessment of the work carried out and a set of endeavors related in particular to the continuation of this work as well as the new research topics which we consider as the most relevant.
Finally, an appendix is associated with our contributions describing the technical implementations to validate the proposals of this thesis.

Chapter 1

Related Work

Table of Contents

1.1 Introduction . . . 9
1.2 Privacy . . . 10
1.2.1 Privacy legislation . . . 11
1.2.2 Privacy design strategies . . . 13
1.2.3 Three-layered privacy model . . . 15
1.2.4 Privacy-preserving architectures . . . 15
1.2.5 Privacy-preserving mechanisms . . . 16
1.2.6 Related privacy-preserving approaches in several domains . . . 19
1.3 Privacy-preserving approaches in the IoT domain . . . 22
1.3.1 Centralized-based approaches . . . 22
1.3.2 Distributed-based approaches . . . 24
1.3.3 Trusted Third-Party-based approaches . . . 28
1.4 Analysis . . . 30
1.5 Summary . . . 33

1.1 Introduction

There is no one universal accepted definition of privacy. According to [Clarke, 2006], four dimensions are considered in order to describe privacy. First, the privacy of personal information that involves the right to control when, where, how, and to whom, data are shared. The second dimension is the privacy of the person that involves the right to control the integrity of the body including the medical devices. The third dimension is the privacy of the personal behavior that involves the right to keep secret any knowledge of the activities and choices. The last dimension is the privacy of the communication that involves the person's right to communicate without surveillance, monitoring or censorship. While the privacy of personal information is the most addressed dimension by privacy laws. However, the other three dimensions are important and

should also be considered in the IoT context because new types of data can be created and communicated between IoT devices. Therefore, we overview in this chapter the privacy notion and define some analysis criteria used to compare the privacy-preserving approaches in the literature. These criteria are based on data protection laws, privacy design strategies, a three-layered privacy model, privacy-preserving architectures, and privacy-preserving mechanisms. Moreover, we survey some existing privacy-preserving approaches in different domains and we compare them. After that, we provide a review of the literature on privacy-preserving approaches proposed in the IoT domain, and we analyze them using our defined analysis criteria.

The remaining of this chapter is structured as follows. We start by giving an overview of the privacy notion in Section 1.2. We then survey the privacy-preserving approaches proposed in the IoT domain in Section 1.3. In Section 1.4, we analyze the presented state-of-the-art before summarizing the content of this chapter in Section 1.5.

1.2 Privacy

Privacy is a complex notion as its definition varies over time, cultures, and among individuals. What one individual considers private is not for another one. The first definition of privacy went back to the 19th century with Warren and Brandeis [Warren and Brandeis, 1890] who defined privacy as "*the right to be let alone*". Then, with the emergence of new technologies, the respect of the privacy became the ability to control personal information. According to Westin [Westin, 1968], privacy is considered as "*the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others*". However, the privacy definition proposed by Westin is argued too general for the IoT area according to Ziegeldorf et al. [Ziegeldorf et al., 2014]. Thus, they proposed a more focused one that defined the IoT privacy as a threefold guarantee including "*(i) awareness of privacy risks imposed by smart things and services surrounding the data owner; (ii) individual control over the collection and processing of personal information by the surrounding smart things; and (iii) awareness and control of subsequent use and dissemination of personal information by these entities to any entity outside the owner's personal control sphere*". Nevertheless, it is important to consider the context when handling the privacy issue in the context of IoT. In fact, data privacy required ensuring data security and taking into account requirements from both the legal regulations and the individual's preferences [Bertino, 2016].

Therefore, we first introduce in this section data protection laws (Section 1.2.1), we then present existing privacy design strategies (Section 1.2.2). We introduce a three-layered privacy model and the privacy-preserving architectures in Section 1.2.3 and Section 1.2.4, respectively. In Section 1.2.5, we present some privacy-preserving mechanisms before surveying some existing privacy-preserving approaches in different domains in Section 1.2.6.

1.2.1 Privacy legislation

The right to privacy is considered as a fundamental human right, enshrined in the United Nations Universal Declaration of Human Rights and Article 8 of the European Convention on Human Rights of 1950 [Pearson, 2009]. In the European Union (EU), this right became explicit by the Directive 95/46/EC [Directive 95/46/EC, 1995], which is based on the OECD guidelines [OECD, 1981]. After that, the General Data Protection Regulation (GDPR) [GDPR, 2016] is adopted by the EU in 2016 and it came into force in 2018 as a successor of the privacy Directive 95/46/EC. Among the main changes between the Directive 95/46/EC and GDPR is the inclusion of the privacy by design notion. In response to the growing importance of privacy by design, the ISO 29100 privacy framework [ISO/IEC29100, 2011] is proposed and defined eleven privacy requirements. However, not every requirement can be met by designing [Hoepman, 2014]. Thus, eight privacy design strategies are defined by Hoepman [Hoepman, 2014] in order to fill the gap between legislation and engineering. These privacy design strategies are detailed on the next Section 1.2.2.

In 1980, the Organization for Economic Co-operation and Development (OECD) issued its Guidelines on the Protection of Privacy and Transborder Flows of Personal Data [OECD, 1981]. The OECD guidelines consisted of eight principles namely, collection limitation, data quality, purpose specification, use limitation, security safeguards, openness, individual participation, and accountability. These principles enabled individuals to express their privacy requirements and place obligations on organizations to follow these requirements. The OECD principles have been the foundations of the EU privacy Directive 95/46/EC [Hoepman, 2014]. For example, Article 6 of the [Directive 95/46/EC, 1995] stated that personal data must be processed fairly and lawfully, must be collected for specified and explicit purposes, and must not be further processed in a way incompatible with these purposes. Moreover, the collected personal data must be adequate, relevant, and not excessive in relation to the initial purposes. Furthermore, the collected personal data must be accurate, up to date, and kept no longer than necessary. All the privacy requirements that are outlined in Article 6 matched the OECD guideline principles, namely purpose specification, collection minimization, and data quality.

As a successor of the Directive 95/46/EC, the European Union adopted the European General Data Protection Regulation (GDPR) [GDPR, 2016] on the protection of individuals with regard to the processing of personal data. For instance, Article 5 of the [GDPR, 2016] covered several OECD guideline principles, such as fairness and transparency (openness in OECD terms), use limitation, and accountability. According to [EU GDPR, 2018], the main changes between the GDPR's privacy principles and the Directive's privacy principles can be summarized in six aspects, including:

- **Consent**: the GDPR strengthens the consent conditions. The consent request must be provided in an easily accessible form. A consent should be clear, distinguishable, and easy to be withdrawn.

- **Breach Notification**: the GDPR makes the breach notification mandatory. The notification

should be sent within 72 hours after being aware of the breach.

- **Right to Access**: the GDPR outlines the data owners' right to be informed on data processing and provided by a copy of the handled personal data.

- **Right to be Forgotten**: the GDPR outlines in Article 17 that data are required to be erased when they are no longer relevant to the original purposes or the consent is withdrawn.

- **Data Portability**: the GDPR outlines the data owners' right to receive their data in a machine-readable format and transmit them to another data controller.

- **Privacy by design**: the GDPR makes the privacy by design as a part of a legal requirement. As outlined in Article 23, the controller must implement appropriate technical and organizational measures in order to meet the requirements of this GDPR and protect the data owners' rights. The main core of the privacy by design is including the data protection at the design stage, rather than an add-on function.

Owing to the privacy by design importance emergence, the International Organization for Standardization (ISO) proposed the ISO 29100 Privacy framework [ISO/IEC29100, 2011]. This framework defines the following eleven privacy safeguarding requirements to protect sensitive information:

- **Consent and choice**: this consists in not only informing the data owner about data collection, but also offering to him/her a choice whether or not to use a data-collecting service.

- **Purpose legitimacy and specification**: this consists in specifying the reasons for collecting personal information.

- **Collection limitation**: this consists in limiting the collection of personal information to what is legal and necessary for a specified purpose.

- **Data minimization**: it aims at minimizing personal information processing.

- **Use, retention and disclosure limitation**: this requirement consists in limiting the use, the retention and the disclosure of personal information to fulfill the specified purpose as long as necessary and thereafter securely destroy the data.

- **Accuracy and quality**: this consists in ensuring that the collected data are accurate, up-to-date and relevant for the purpose.

- **Openness, transparency and notice**: this consists in providing clear, complete and accessible information on personal information processing.

- **Individual participation and access**: this consists in accessing to the personal information and correcting inaccuracies.

- **Accountability**: it consists in reporting on personal information.

- **Information security**: this consists in protecting personal information under the data owner authority with the appropriate control.

- **Privacy compliance**: it consists in verifying and demonstrating adherence to laws with internal or third-party audits.

The ISO 29100 Privacy framework [ISO/IEC29100, 2011] presents several aspects with the intention to enhance the privacy protection. However, not every requirement can be met by designing [Hoepman, 2014]. Thus, translating the privacy principles described above to more engineer-friendly principles are required in order to facilitate their implementation. In 2014, Hoepman [Hoepman, 2014] defined eight privacy design strategies that are detailed on the following Section 1.2.2.

1.2.2 Privacy design strategies

Hoepman [Hoepman, 2014] derived eight privacy design strategies from the privacy guidelines [OECD, 1981], data protection laws [Directive 95/46/EC, 1995] [GDPR, 2016], as well as privacy framework[ISO/IEC29100, 2011]. These strategies are defined as below [Hoepman, 2014]:

- **Minimize**: the personal data that are collected, stored and disseminated should be restricted to the minimal amount possible. This strategy matches the data minimization legal principle. Select a subset of the incoming data, anonymization and use pseudonyms can implement this strategy [Pfitzmann and Köhntopp, 2001].

- **Hide**: any personal data and their interrelationships should be hidden from plain view. This strategy matches the data minimization legal principle. In certain cases, this strategy can be used to hide information from anybody. Thus, differential privacy [Mivule, 2013], anonymization and use pseudonyms can implement this strategy in order to achieve unlinkability [Pfitzmann and Köhntopp, 2001]. In other cases, this strategy's intent is to hide the information from other parties excepting the legitimate party. Thus, encryption can be used to ensure the confidentiality.

- **Separate**: personal data should be processed in a distributed fashion, in separate compartments whenever possible. This strategy calls for distributed data storage and processing instead of centralized solutions. This strategy matches the purpose limitation legal principle.

- **Aggregate**: personal data should be processed at the highest level of aggregation and with the least possible detail. This strategy matches the data minimization legal principle. K-anonymity [Sweeney, 2002], l-diversity [Machanavajjhala et al., 2006], and t-closeness [Li et al., 2007] can implement this strategy.

- **Inform**: data owners should be adequately informed whenever personal data are processed. The notifications include which information are processed, why, and how they are

protected. This strategy matches the transparency, right to access, and breach notification legal principles.

- **Control**: data owners should be provided by the appropriate solutions to control their personal data. Data protection rights need to be implemented in order to enable the data owner to view, erase, and rectify personal data. This strategy matches the right to access and data portability legal principles.

- **Enforce**: a privacy policy compatible with legal requirements should be in place and should be enforced. Thus, technical protection mechanisms (e.g., access control) need to be in place to prevent privacy policy violations. This strategy matches the purpose limitation, data quality, right to be forgotten, and information security legal principles.

- **Demonstrate**: this requires a data controller to be able to demonstrate compliance with the privacy policy and any applicable legal requirements. The use of logging and auditing can implement the demonstrate strategy. This strategy matches the accountability legal principle.

Table 1.1 shows the legal principle coverage by the eight privacy design strategies. Each privacy legal principle is matched by at least one of the design strategies. Thus, the privacy design strategies can be used in order to evaluate the privacy-preserving approaches proposed in the literature in terms of privacy framework, law, and regulation compliance.

Table 1.1: Mapping of privacy design strategies onto privacy legal principles

	Data minimization	Purpose limitation	Data quality	Transparency	Right to Access	Breach notification	Data portability	Right to be Forgotten	Information security	Accountability
Minimize	✓									
Hide	✓									
Separate		✓								
Aggregate	✓									
Inform				✓	✓	✓				
Control					✓		✓			
Enforce		✓	✓					✓	✓	
Demonstrate										✓

The privacy design strategies need to be applied by both the data owners and the data consumers in order to preserve the data owner's privacy and prove the privacy legal requirement compliance by the data consumers. Hence, we split these strategies on a three-layered privacy model on the following section.

1.2.3 Three-layered privacy model

Preserving privacy is a collaborative task that needs the implication of all the involved parties in order to protect the handled data during the collection, transmission, storage and the processing phases. According to Spiekermann and Cranor [Spiekermann and Cranor, 2009], we can distinguish three areas where privacy needs to be preserved, namely the **user sphere**, where the user's data are fully controlled by the owner, the **joint sphere**, where both the data owner and the data consumer control the user's data, and the **recipient sphere**, where the user's data are out of the data owner control.

In order to enable privacy in engineering and ensure an end-to-end privacy preservation, data need to be carefully protected in each sphere. In this context, we split the privacy design strategies on the three-layered privacy model, as below:

- **User Sphere**: in this area, the user's devices and data are completely controlled by their owner. Minimizing data locally on the user sphere can enhance privacy preservation more than trying to prevent the data consumer from analyzing raw data that already possessed. Thus, we argue that applying the first four privacy design strategies, which are *minimize*, *hide*, *separate*, and *aggregate* before outsourcing the data from the owner's control area is more efficient.

- **Recipient Sphere**: in this area, the user's data are out of the owner control because they are transferred to a third-party that is the only controller. According to Article 5 of the GRPD, some measures need to be respected in order to ensure lawful and fair processing. Thus, by adopting the last four privacy design strategies, which are *inform*, *control*, *enforce*, and *demonstrate*, the data consumer can prove its privacy legal requirement compliance to the data owner or the data controller.

- **Joint Sphere**: in this area, the user's data are hosted by third-party that provides services (e.g., e-mail). Both the data owner and the data consumer share the control over the user's data. Thus, all the privacy design strategies need to be adopted in this area. Thus, the data owner adopts the first four privacy design strategies and the data consumer adopts the last four privacy design strategies.

In the next section, we present the privacy-preserving architectures.

1.2.4 Privacy-preserving architectures

In order to preserve privacy, we distinguish three types of architecture that are: centralized, distributed, and third-party architectures.

- **Centralized architecture (Cen)**: this requires a local central node that resides at the user sphere and preserves the collected data privacy before that the data become out of the data owner's control. The main challenge with this architecture is that all the computation tasks are managed by a single server. Thus, the data owner's privacy can be threatened in case of server hacking.

- **Distributed architecture (Dis)**: this requires that all the entities in the network collaborate among them in order to protect their data privacy. Although this architecture overcomes the single point of failure, malicious entities intrusion arises because any entity can connect with any other entity at any time.

- **Trusted Third-Party architecture (TTP)**: this requires a trusted third-party, which can be a cloud provider, a data collector, a consumer, a public institution, or a private corporation that is responsible for preserving privacy during the data collection, transmission, storage, and/or processing. The third-party is an external server from the user sphere. The main challenge with such architecture is that it gives full trust to the third-party for the whole data protection.

After presenting the privacy-preserving architectures, we introduce in the following section the privacy-preserving mechanisms.

1.2.5 Privacy-preserving mechanisms

Existing privacy-preserving mechanisms are based on perturbation, restriction, aggregation, semantic, and blockchain.

1.2.5.1 Perturbation-based mechanisms

These mechanisms rely on a series of operations that modify or hide some sensitive parts on the original data to preserve privacy [Fang et al., 2016]. To this end, noise addition and anonymization techniques are adopted.

Noise addition techniques: These techniques transform confidential attributes by adding noise to the original data to prevent the identification of a particular individual [Mivule, 2013]. They can be categorized into four groups: (1) data sampling techniques, which aim at releasing a new table that includes only the data of a sample for the whole population, (2) random-noise techniques, which consist of adding or multiplying the value of the sensitive attribute with a randomized number, (3) data swapping techniques, which modify a subset of the data by introducing uncertainty about the true data value [Sharma et al., 2013], and (4) differential privacy techniques, which consist of adding Laplace noise to a database query result [Mivule, 2013].

Anonymization protection techniques: These techniques hide a data owner's identity by removing any explicit identifier and makes the data less precise. There are three well-known privacy-preserving methods: k-anonymity [Sweeney, 2002], l-diversity [Machanavajjhala et al., 2006], and t-closeness [Li et al., 2007]. The k-anonymity is a formal method that is proposed to counter the re-identification problem caused by the quasi-identifier attributes. However, k-anonymity can be susceptible to background knowledge attacks. Therefore, researchers designed other versions, such as l-diversity [Machanavajjhala et al., 2006] whose main idea is that there must be at least l distinct values for the sensitive attribute in each quasi-identifier group as well as t-closeness method [Li et al., 2007], which requires the distribution of a sensitive attribute in any quasi-identifier group to be close to the distribution of the attribute in the overall table.

1.2.5.2 Restriction-based mechanisms

These mechanisms aim at limiting data use by blocking access or encrypting inputs. Data restriction methods include access control and cryptography-based techniques.

Access Control: These techniques are effective for ensuring data sharing [Fang et al., 2016]. Data owners can express their individual preferences about who can access what data and how others manipulate their shared data. Control mechanisms include Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC). RBAC assigns access permissions based on the roles whereas ABAC defines permissions based on attributes, such as subject, resource, and environment attributes [Fang et al., 2016].

Cryptographic protection: These techniques are heavily used when preserving privacy. They can be categorized into two major groups, namely: asymmetric/symmetric encryption and public key infrastructure. The asymmetric/symmetric encryption uses keys to protect the data. While the public key infrastructure delivers the entity a certificate to make sure that the public key belongs to the identified entity.

1.2.5.3 Aggregation-based mechanisms

These mechanisms aim at releasing information about the data provided by several users, without any information leakage about individual data. Data aggregation methods include multiparty computation and homomorphic encryption-based techniques.

Multiparty computation: These techniques are effective for ensuring data computation by relying on multiple parties that have to cooperate to get the aggregated result without the involvement of any trusted party. Different solutions for distributing data among the computation sites exist, based on logic circuits [Yao, 1986], arithmetic circuits [Ben-Or et al., 1988], or linear secret sharing schemes [Cramer et al., 2000]. Solutions based on linear secret sharing, including Sharemind [Bogdanov et al., 2008], SEPIA [Burkhart et al., 2010], P4P [Youdao, 2010] and [Iacovazzi et al., 2013], have been demonstrated to effectively scale to large numbers of users [Randazzo et al., 2015].

Homomorphic encryption-based protection: These techniques are heavily used when preserving privacy such that they enable obtaining computation results over ciphertext calculation without knowing the appropriate plaintexts and private keys of the ciphertexts [Acar et al., 2018]. There are several homomorphic encryption schemes in the literature, such as RSA [Rivest et al., 1978], ElGamal [ElGamal, 1985], and Paillier cryptosystem [Paillier, 1999]. According to [Acar et al., 2018], both RSA and ElGamal cryptosystems are only multiplicatively homomorphic. Hence, they do not allow the homomorphic addition of ciphertexts. However, the Paillier cryptosystem implements the additive and multiplication operations.

1.2.5.4 Semantic-based mechanisms

These mechanisms aim at representing a domain in a standard format using a common notation in order to overcome the heterogeneity issue in a given domain. Two goals can be ensured by the

ontological modeling techniques, namely domain description and inference-based reasoning.
Domain description: ontological modeling techniques are known as successful techniques in representing knowledge [Hosseinzadeh et al., 2016]. The Resource Description Framework (RDF), the RDF Schema (RDFS), and the Web Ontology Language (OWL) are the well-known languages used to implement ontologies. While RDF and RDFS are used to define the structure of the data, such that expressing a term's type using the *rdf:type* predicate or expressing a relationship between two classes, OWL adds semantics to the schema, such that expressing the *owl:sameAs* relationship between two terms. Moreover, OWL language provides a formal representation enabling the check of inconsistencies, the visualization of the ontology structure, and the ease of maintenance.
Inference-based reasoning: some relationship cannot be expressed in OWL. Thus, a set of inference rules need to be defined in order to interrogate the ontology and generate new knowledge built upon the ontology's terms expressed on OWL language. One of the well-known inference languages is the Semantic Web Rule Language (SWRL). Indeed, SWRL language provides easy support for the integration of inference over ontology's terms. For instance, by modeling the users' privacy preferences as a semantic-based privacy policy, automated reasoning engines can interpret these policies and automatically infer whether or not a particular data use is compliant with the privacy policy [Kagal and Pato, 2010].

1.2.5.5 Blockchain-based mechanisms

These mechanisms aim at overcoming the problem related to trusting a centralized party in order to meet the privacy requirements. Two blockchain types exist, namely permissionless and permissioned blockchains [Kshetri, 2017].
Permissionless blockchain: The permissionless blockchain is an open platform that anyone can join it. The first proposed system was Bitcoin [Nakamoto et al., 2008], which allows users to transfer securely the currency (bitcoins) without a centralized regulator. Specific nodes in the network known as miners are responsible for collecting transactions, solving challenging computational puzzles (proof-of-work) in order to reach consensus and adding the transactions in form of blocks to a distributed public ledger known as the blockchain. Since then, other projects demonstrate how these blockchains can serve in other domains, such as the Storj project [Storj, 2014], which is a decentralized peer-to-peer cloud storage network, and the Onename project [Onename, 2016], which is a distributed and secured identity platform.
Permissioned blockchain: The permissioned blockchain is restrictive and required some authority that granted authorized accesses. For instance, Hyperledger Fabric [Androulaki et al., 2018] is an open source enterprise-grade permissioned distributed ledger technology platform proposed by IBM. The idea behind the permissioned blockchains is that the problems faced by permissionless blockchains can be avoided by controlling access to only trusted users on the platform. Permissioned blockchain is also used in order to address the privacy issue in the IoT domain, namely the Ancile solution [Dagher et al., 2018], which is a blockchain-based framework for secure and efficient share the patient's medical records.

Based on the above overview of privacy, we survey some existing privacy-preserving approaches in different domains in the following section.

1.2.6 Related privacy-preserving approaches in several domains

In this section, we present some privacy-preserving approaches in order to provide a thorough view of privacy in different domains. However, they are not direct solutions to our research problem. Therefore, we do not consider them as competitors because they do not support the IoT field.

Traditional security and privacy mechanisms tailored to traditional data management systems are inadequate for Big Data [Colombo and Ferrari, 2015]. In this context, Colombo and Ferrari proposed a first step to integrate privacy-aware access control features into existing big data platforms in order to deal with the relevant threats to privacy implied by the big data analysis. Thus, the authors presented the foundations of a framework for the integration of privacy-aware access control (PAAC) into MapReduce systems and NoSQL datastores. They discussed a variety of activities, such as the identification of policies for BigData platforms, the definition of policy specification, and the definition of enforcement mechanisms. Moreover, the query rewriting techniques can be used to the distributed processing capabilities of host Big Data platforms.

For big data storage, Liang at al. [Liang et al., 2015] proposed an anonymous identity-based conditional proxy re-encryption scheme. The proposed privacy-preserving ciphertext multi-sharing mechanism combined the merits of proxy re-encryption with anonymous techniques in order to share a ciphertext multiple times by updating the ciphertext recipient without leaking both the knowledge of the plaintext and the identity information of ciphertext senders and recipients. However, the proxy can create delegation rights between the two parties.

In order to share sensitive data in the healthcare domain, Yang et al. [Yang et al., 2015] proposed a solution for medical record sharing for cloud computing while preserving the patient's privacy. Vertical partition of the medical dataset is used in order to achieve the consideration of different parts of medical data with different privacy concerns. The original medical dataset is vertically partitioned into three parts for the remote storage in the cloud. The part that can be used for patient identification is stored in ciphertext. The other parts that will be used for medical analysis are stored in plaintext. The cloud environment is used as a joint sphere to share the medical records. Both SSL and TSL are used for privacy protection during transmission among the three parties namely, the data owner, the cloud service provider, and the data recipient. This proposal needed a complex process to protect data privacy, and the recipient had to interact with the data owner to access the original dataset.

Privacy-preserving data mining refers to aggregate information provided by several users without any information leakage about individual data [Randazzo et al., 2015]. In this context, Randazzo et al. considered the approach of multiple sites that need to cooperate to get the mining results. Thus, they relied upon the Shamir's secret sharing scheme and take advantage of its homomorphic property. A central unit is assumed to monitor the aggregate behavior

of the users. Finally, the authors experimentally evaluated two promising privacy-preserving techniques. In the first scheme, each node sent its secret data to every network's node, whereas in the enhanced scheme, each node sent its secret data to a set of chosen nodes.

Different from the widely used cryptographic approaches, He et al. [He et al., 2016] addressed the privacy-preserving data aggregation in ad hoc networks problem by exploiting the distributed consensus technique. Indeed, the authors proposed a secure consensus-based data aggregation algorithm, called SCDA that guarantees an accurate sum aggregation while preserving the privacy of sensitive data. According to the authors, SCDA did not rely on a centralized controller or a trusted aggregator, and it could be implemented in a distributed manner and robust against the network dynamics. However, this work did not deal with the permanent node failure issue.

The creation of collaborative working environments became easier thanks to the availability of current networking and computing power [Allison et al., 2016]. Hence, Allison et al. proposed a framework that aimed at preserving privacy while collaborating. The proposed framework contained a generic privacy ontology, a reasoning engine, and a collaborative privacy manager. The privacy ontology allowed the framework to adapt to any domain. The reasoning engine role is to infer who had access to what information according to which privacy rules. The collaborative privacy manager aimed at making decisions, assisting collaborating users with their privacy preferences, and providing users with knowledge and suggestions.

Ontology is also used for preserving privacy in a ubiquitous computing environment. Bawany and Shaikh [Bawany and Shaikh, 2017] proposed a Privacy Manager, which is an ontology-based solution for data privacy. The basic idea is that the data producers specified a set of privacy requirements, which the data consumers must satisfy in order to access the requested data. The Privacy Manager enabled (i) the data producers to define their privacy policies and (ii) the data consumers to access the requested data only if the access is allowed by the defined privacy policy. The ontology included two privacy policy types, namely *TimePrivacyPolicy* that restricted the access duration and *LocationPrivacyPolicy* that restricted the access location, including physical or virtual location.

According to Froelicher et al. [Froelicher et al., 2017], using a centralized authority to preserve privacy while data sharing among multiple parties is not an appropriate solution. For that, Froelicher et al. proposed a decentralized system, called UnLynx for efficient privacy-preserving data sharing. UnLynx not only guaranteed the confidentiality and the unlinkability between data providers and their data, but also preserved the end result privacy and the correctness of computations by the servers. UnLynx is based on homomorphic cryptography, verifiable shuffling, and zero-knowledge proofs. Both the data and the computations are decentralized in UnLynx. Thus, no central repository for storing all data neither a central authority responsible for all the computations.

More recently, the blockchain has been proposed as a possible solution for managing privacy-preserving. Using this technology, Liang et al. [Liang et al., 2017] proposed a decentralized and trusted cloud data provenance architecture, called ProvChain. ProvChain collected and

verified cloud data provenance by embedding the provenance data into blockchain transactions. ProvChain recorded the operation history as provenance data which will be hashed into Merkle tree nodes. In ProvChain, only the hashed identity of users are kept to protect their privacy in the blockchain network. Concerning the rest of the user data, the authors assumed that they are encrypted and stored on the cloud and are not accessible to anyone without the decryption key.

Traditional centralized crowdsourcing system suffered from privacy disclosure, single point of failure, and high service fee [Li et al., 2018]. For this reason, Li et al. proposed a blockchain-based decentralized framework for crowdsourcing, called CrowdBC. A requester's task can be solved by a crowd of workers without relying on any trusted third institution while guarantying the users' privacy. In traditional crowdsourcing systems, user's sensitive information (e.g. name, email address, and phone number) and task solutions are saved in the database of crowdsourcing systems, which had the risk of privacy disclosure and data loss. CrowdBC overcame this issue by using the pseudonymous Bitcoin-like addresses to identify requesters and workers, which enables privacy-preserving without submitting true identity to finish a crowdsourcing task.

In summary, we classify the surveyed privacy-preserving approaches and their categorization in Table 1.2.

Table 1.2: List of the privacy-preserving approaches in several domains

		User Sphere				**Joint Sphere**		**Recipient Sphere**			
	Architecture	Minimize	Hide	Separate	Aggregate	Secure Communication	Anonymous Communication	Inform	Control	Enforce	Demonstrate
Perturbation-based mechanisms											
[Yang et al., 2015]	TTP		✓	✓	✓	✓			✓	✓	
[Froelicher et al., 2017]	Dis		✓	✓	✓	✓				✓	
Restriction-based mechanisms											
[Colombo and Ferrari, 2015]	TTP			✓		✓				✓	
[Liang et al., 2015]	Dis		✓				✓			✓	
Aggregation-based mechanisms											
[Randazzo et al., 2015]	Dis	✓	✓		✓	✓					
[He et al., 2016]	Dis	✓	✓		✓	✓					
Semantic-based mechanisms											
[Allison et al., 2016]	Cen	✓				✓		✓		✓	
[Bawany and Shaikh, 2017]	TTP	✓				✓		✓	✓	✓	
Blockchain-based mechanisms											
[Liang et al., 2017]	Dis		✓				✓		✓	✓	✓
[Li et al., 2018]	Dis		✓				✓				

Some of these approaches enabled preserving privacy by trusting a third-party [Yang et al., 2015] [Colombo and Ferrari, 2015] [Bawany and Shaikh, 2017], while some other solutions are

rather distributed and eliminated the need for trust [Liang et al., 2015] [Liang et al., 2017] [Li et al., 2018]. The *inform* privacy design strategy is only covered by Semantic-based approaches [Allison et al., 2016] [Bawany and Shaikh, 2017], in which the data owner is enabled to select the collected data and is notified when data are processed. Besides, the *control* strategy is partially ensured in [Yang et al., 2015] and [Liang et al., 2017] by proposing for the data owner a data integrity checker and a key to decrypt and modify the stored data, respectively.
It is worth noting that these solutions addressed different problems than ours because they do not consider privacy for the IoT data while considering the low computation and energy capacity of the IoT devices. Thus, all of these solutions used traditional privacy and security techniques that faced several challenges in the constrained environment, namely the IoT domain.

After presenting some privacy-preserving approaches in several domains, we survey on the following section more specific approaches proposed in the IoT domain.

1.3 Privacy-preserving approaches in the IoT domain

In this section, we provide a review of the literature on privacy-preserving approaches proposed in the IoT domain. We categorize them based on the three architecture types defined on Section 1.2.4.

1.3.1 Centralized-based approaches

Centralized-based approaches rely on one or few central nodes in which data are stored and/or aggregated in order to preserve the privacy before sending the data to the data consumers.

Lai et al. [Lai et al., 2014] proposed a conditional privacy-preserving authentication with access linkability for roaming service, called CPAL. The proposed CPAL scheme can achieve session key agreement, strong anonymous authentication, and fast user tracking. Moreover, CPAL provided anonymous user linkability for roaming service in order to provide universal secure roaming service and multilevel privacy preservation. The authors used a group of signature techniques, which enabled the authorized entities to link all the access information of the same user without knowing the user's real identity. The authorized foreign service providers can use the master linking key possessed by the trust linking server in order to link the access information from the user to improve their services while preserving user anonymity.

For their part, Kravets et al. [Kravets et al., 2015] proposed a secure and privacy-preserving IoT framework, called Incognito where users can create identities that work only within certain contexts and are meaningless outside of these contexts. Thus, the user can generate a new identity for each given context and choose the information exposure level. Incognito preserved the individuals' privacy by giving them full control over the information traces that they leave behind in an IoT infrastructure. Each user had a pool of pseudonyms to manage their information exposure in a given context. Incognito set the BLE MAC address of a user's device to a temporary identity-based ID. Nonetheless, these temporary identifiers are used for long enough that short-term shadowing and tracking can still be a problem. Besides, a system component is

installed on the user's smartphone or smartwatch in order to manage the user device identities. However, the IoT devices are in general resource-constrained and can be attacked.

In order to address the problem of involuntary privacy breaching risk minimization in smart energy management systems, Ukil et al. [Ukil et al., 2015] proposed a solution, called Dynamic Privacy Analyzer (DPA). The proposal managed privacy by detecting, measuring, and preserving the smart meter data privacy before they are shared with third-parties. DPA received smart meter data through secure channels and used the Internet to send alerts to the user's smartphone and privacy preserved data to third-party applications. While preserving privacy minimized the risk of privacy breaches for data owners, it led to loss of sensor data quality that may be useful for the data consumers.

For their part, González-Manzano et al. [González-Manzano et al., 2016] proposed a Privacy-preserving Aggregation protocol, called PAgIoT suitable for resource-constrained IoT devices. PAgIoT enabled multi-attribute aggregation for a group of entities while allowing for privacy-preserving value correlation. Moreover, the aim of the proposed mechanism were the privacy preservation, the collision resistance, and the correlative aggregation. In PAgIoT, data are decomposed into a set of attributes, which are aggregated separately. Hence, a central node (sink) queried for the value of certain attributes and the remainder nodes responded depending on whether they possessed or not these attributes.

In order to enable information security and privacy protection for Smart Spaces, Hosseinzadeh et al. [Hosseinzadeh et al., 2016] proposed a security framework, which is composed of various components collaborating together to support different aspects of security, such as authentication, authorization, and access control. Moreover, a context-aware role-based access control scheme is proposed, which is modeled using the ontological techniques and the Web Ontology Language (OWL) and implemented by the C Language Integrated Production System (CLIPS) rules. In this model, roles are assigned to the users by the administrator when they registered in the system. At run-time, the privacy rules are executed to grant or deny access based on the user's role and the context information. By activating roles, users got the rights to access the data. The rights are permissions or prohibitions to perform four actions, namely *read*, *update*, *delete*, and *query* data.

Wang et al. [Wang et al., 2016] proposed an Ontology-based Resource Description Model, called ORDM in order to describe resources in the IoT environment, which are described by the attribute, state, control, historical information and privacy classes. The Attribute class defined the inherent information of the device, such as the device type, model, and range of the sensed values. The data description is made in the State class, which provided the current data captured by the sensor with their associated data unit. The Privacy class protected the device from illegal access or control. A smart office application based on the ORDM is implemented for the evaluation. However, ORDM did not offer fine-grained access control to the sensed data. Indeed, the users that can access the IoT resource are fixed in the proposed ontology without any reasoning or clear criteria. Moreover, the authors did not deal with data resource sharing during the data processing phase. Furthermore, only access and control authorizations are considered

as privacy requirements. ORDM did not consider the rest of the privacy requirements, such as purpose specification, retention, and disclosure limitation.

Lu et al. [Lu et al., 2017] proposed a lightweight privacy-preserving data aggregation scheme, called LPDA for fog computing-enhanced IoT. LPDA employed the one-way hash chain techniques in order to enable a fog node to filter false data by running the source authentication at the network edge. Besides, LPDA combined the homomorphic Paillier encryption and the Chinese Remainder Theorem to aggregate hybrid IoT devices' data into one ciphertext. LPDA included four actors, namely IoT devices, a fog device, a control center, and a trusted authority. A set of IoT devices periodically reported their data to a fog device, which aggregated the received data and forwarded them to the control center that can make data analytics over the aggregated data. The trusted authority's role is to assign and manage keys to all the IoT devices, the fog node, and the control center.

Guan et al. [Guan et al., 2018] proposed a privacy-preserving and efficient data aggregation scheme based on the blockchain to preserve the user's privacy in a smart grid. The proposed scheme consisted of three principal entities, namely smart meters that sent their consumption data to a mining node to be aggregated, a center unit that received the aggregated data from each group, and a key management center that is responsible for initializing the users' keys. Smart meters are divided into different groups and each group has a private blockchain to record the participants' data. For data aggregation, one smart meter, called mining node is randomly chosen to aggregate the group's members' data and record the aggregated data into the group's private blockchain. The user's identity is hidden using the pseudonymity in order to preserve the inner privacy within a group.

Guan et al. [Guan et al., 2019] proposed a device-oriented anonymous privacy-preserving scheme with authentication, called APPA in order to address privacy-preserving data aggregation in the fog-enhanced IoT environment. APPA scheme consisted of five entities, namely smart devices, fog node, public cloud server, trusted certification authority, and local certification authority. The two authorities are independent agencies responsible for the system's certification management. Smart devices collected and sent the data to the fog node that stored and aggregated the received data using the Paillier cryptosystem. Then, it forwarded the aggregated data to the public cloud server that processed the data to better serve the users. Moreover, both the anonymity and authenticity of the device are guaranteed with a pseudonym and a pseudonym certificate.

The main challenge with the centralized-based approaches is that all the preserving privacy computation tasks are managed by a single node. Thus, in a case of node hacking, all the user's sensitive data are attacked.

1.3.2 Distributed-based approaches

In order to overcome the single point of failure issue, the distributed-based approaches enabled all the network entities to collaborate among them in order to protect their data privacy.

In the IoT domain, addressing the privacy issue can be through a device-centric perspective

by focusing on issues related to the IoT device capabilities. In order to manage the IoT device provenance, Hardjono and Smith [Hardjono and Smith, 2016] proposed a privacy-preserving method for commissioning an IoT device into a cloud ecosystem. Their new ChainAnchor architecture helped an IoT device to prove its manufacturing provenance without the authentication of a third-party and it is allowed to register anonymously through the use of a blockchain system. In this paper, the Enhanced Privacy ID (EPID) zero knowledge proof scheme is used to achieve and prove the participants' anonymity and membership.

Besides the device-centric perspective, several solutions addressed the IoT privacy through a data-centric perspective by focusing on issues related to data aggregation, data storage, and data analysis. For instance, Wong and Kim [Wong and Kim, 2014] proposed a self-awareness data collection protocol to raise the confidence of the data owners when submitting their personal data to the data collector. The proposal privacy preserved approach enabled the data owners to learn about the anonymous protection level (e.g., k-anonymity) that is proposed by the data collector before the data submission. The proposed protocol required that data owners helped each other in preserving each data owner's privacy. The Tor network is used to prevent direct communication between the data collector and the data owners. Thus, the data collector cannot track the identity of any data owner. All the data are stored on the data collector after applying the appropriate protection level. Hence, the owner lost the control over its outsourced data.

Birman et al. [Birman et al., 2015] proposed a design for a smart metering system that allowed utilities to use the collected data effectively while preserving the privacy of the data owners. The system operator (i.e., the utility) is treated as an honest-but-curious adversary, which ran the system correctly but cannot be trusted with access to data it did not need to know. Three layers are defined namely, the communication layer, the data mining layer, and the control layer. According to the authors, keeping data on the consumer's device is better than trying to prevent the utility from reading or analyzing data that already possessed. Although keeping data on the IoT devices can improve preserving privacy in the IoT domain, the smart meters had not sufficient resources to convert and aggregate the data before sending them to the utility. These tasks were difficult and costly in terms of energy consumption, such as encryption of the messages exchanged between the meters, noise addition to the meter data, etc.

Jayaraman et al. [Jayaraman et al., 2017] proposed innovative techniques for privacy preservation of IoT data, which used multiple IoT cloud data stores to protect the IoT data privacy. The aim of the proposed technique was to decompose the IoT data to attends, store them in multiple data stores, and re-aggregate them when asked by a consumer without exposing anything beyond meaningless addends. The solution avoided the single point of attack/failure issue thanks to the distributed computing and the use of homomorphic properties of the Paillier cryptosystem that allowed analyzed IoT data retrieval without exposing the raw data.

For their part, Abdallah and Shen [Abdallah and Shen, 2018] proposed a lightweight lattice-based homomorphic privacy-preserving electricity consumption aggregation scheme. The proposed network model consisted of one control center that is connected to several base stations. Each base station is responsible for a cluster of home area networks; each network included

one smart meter and several smart household appliances. An independent trusted authority provided keys. According to the authors, their proposed scheme allowed the smart household appliances to aggregate their data among them without the smart meter or the base station involvement. Nevertheless, both the smart meter and the base station can check the encrypted consumption message's authenticity without recovering their contents by verifying the message's signature.

The above study relayed on a trusted authority for key management. In order to eliminate such trusted third-party, Liu et al. [Liu et al., 2018] proposed a practical privacy-preserving data aggregation scheme, called 3PDA, in which the users constructed a virtual aggregation area to mask the single user's data. The proposed scheme consisted of several smart meters that collected electricity consumption data, a data collection unit that aggregated the data in an aggregation area, and an operation center that received the aggregated result. In order to eliminate the trusted third-party, the set of smart meters in an aggregation area needed to collaborate among them in order to generate a group key used to encrypt each meter's data and decrypt the aggregated data computed by the data collection unit. Moreover, the computational Diffie-Hellman is used to achieve the confidentiality, as well as the authentication and integrity are verified using the digital signature. However, eliminating the trusted key management authority required additional computation to be executed by the smart meters to generate a group key.

Due to the overhead of the fully homomorphic encryption and secure multiparty computation, Tonyali et al. [Tonyali et al., 2018] adapted these two systems to be deployed in smart grid advanced metering infrastructure networks that are formed using wireless mesh networks. To this end, the authors proposed to add a presentation layer to include packet size information at the sender size in order to reduce the large ciphertext size and deal with the packet reassembly problem. Besides, they proposed a communication protocol in order to reduce the message complexity. Moreover, the digital signature is used to achieve the message authentication and the data integrity.

More recently, the use of blockchain for applications outside of currency and financial services has also received significant attention. Zyskind et al. [Zyskind et al., 2015] proposed a decentralized personal data management system in order to ensure that users owned and controlled their personal data. The authors combined blockchain and off-blockchain storage to construct an access control manager focused on privacy. The main role of the proposed blockchain was enforcing access policies that defined which data can be shared and with whom. Each entity in the proposed system is represented by a public key and a policy that specified restricted accesses for the public keys of the interested entities. Policies are stored on the blockchain and only the user is allowed to change them. The blockchain nodes verified whether or not these policies are respected. Moreover, a trust measure that is based on node behavior is used to give more weight to trusted nodes in mining.

For their part, Hashemi et al. [Hashemi et al., 2016] proposed a decentralized solution for sharing data in the IoT environment that consisted of a distributed data storage system. The proposed system used blockchain to maintain the data access control and the data storage model.

The main features of this system were (i) separation of the data store and the data management. This latter is ensured using the blockchain technology, (ii) a decentralized access control, and (iii) a scalable messaging based on publish/subscribe model to query data. However, the authors assumed that the IoT devices had sufficient resources in order to solve the Proof-Of-Work that might not always be true. In fact, solving the POW for Bitcoin required very sophisticated hardware.

Ouaddah et al. [Ouaddah et al., 2016] proposed a new framework, called FairAccess for access control in IoT based on the blockchain technology. The authors reused the Bitcoin system and introduced some new types of transaction, such as *grant*, *get*, and *revoke* access. Moreover, the model included three actors, namely the shared resource, the resource owner, and the users. The transactions are used to provide access control, and the blockchain is used for storing and reading the permissions. The resource owner controlled the resource accesses through transactions. For instance, the resource owner can create a grant access transaction that specified a user who had the right to access the shared resource. Once one miner included this transaction in the blockchain, the user can directly access the resource using the received access token.

Shafagh et al. [Shafagh et al., 2017] introduced an initial design of a blockchain-based data storage system. The authors enabled a secure and persistent data management by using the blockchain as an auditable access control layer to a decentralized storage layer, named data plane. This latter is based on the off-chain Decentralized Hash Table (DHT) technology, which stored key-value pairs. The value was the data chunk, and the key was the resulting hash value of the following tuple: <stream-ID, owner-ID, timestamp-hash>. The authors proposed to store data chunks, which composed several consecutive data records instead of storing data records. To this end, the authors split a data stream into several data chunks, which are cryptographically chained together (i.e., each chunk held a hash pointer to the previous chunk). Before a chunk left the source, it is encrypted with a symmetric cipher. Moreover, access rights are granted per data stream and are limited in time. Hence, when a DHT node received a data access request, it interacted with the blockchain to check whether or not the requester can retrieve such data.

Dorri et al. [Dorri et al., 2017a] proposed a lightweight and optimized blockchain for resource-constrained devices. The proposed solution eliminated the overhead associated with the classic blockchain. The proposed blockchain eliminated the mining and thus did not incur additional delays in processing generated transactions. Under the proposed model, only authorized users can access and control devices in the house. Besides, messages received by the authorized users are protected and cannot be modified by any malicious users. In [Dorri et al., 2017b], the same authors applied their lightweight blockchain for a smart house. In each house, multiple IoT devices (e.g., smartphones, personal desktops, and sensors) are connected to the same network. In addition, each house is equipped with an online, powerful resource device, and known as the house's miner. This latter is responsible for handling all transactions inside the house.

For their part, Wang et al. [Wang et al., 2018] proposed a data aggregation framework to aggregate and verify meter data by a hierarchical blockchain system, in which the consensus mechanism is supported by the practical byzantine fault tolerance algorithm [Castro et al., 1999].

The proposed scheme consisted of three principal entities, namely smart meters, cluster gateways, and a substation. Smart meters in the demand side are grouped as clusters based on their geographical locations. Each cluster is equipped with a cluster gateway that aggregated the smart meters' data and then forwarded the aggregated data to the substation. Thus, the additive homomorphic technique enabled the substation to track regional energy consumption without revealing the plaintext of meter data.

For the healthcare domain, Dagher et al. [Dagher et al., 2018] proposed a blockchain-based framework, called Ancile for secure and efficient access to medical records by patients, providers, and third-parties, while preserving the patients' privacy. Ancile employed smart contracts, data obfuscation techniques, and cryptographic techniques in order to improve privacy and security in the healthcare domain. Moreover, Ancile blockchain stored hashes of the data references while sending the actual query link information in a private transaction over HTTPS. The permissioned blockchain structure is based on a consensus algorithm rather than a proof of work.

Although the distributed-based approaches are privacy-friendly solutions, malicious entities intrusion arose because any entity can connect with any other entity at any time.

1.3.3 Trusted Third-Party-based approaches

In order to address the malicious entities intrusion rise issue, some sensitive computation tasks can be delegated to one or several trusted third-parties.

Huertas Celdrán et al. [Huertas Celdrán et al., 2014] proposed a middleware, called Privacy-Aware Recommender Based on Context Information for Cloud Service Environments (PRECISE) that provided users with context-aware recommendations by considering the context information, the users' locations, the privacy policies, and the previously visited places. PRECISE and the context-aware services are allocated at the platform as a service (PaaS) and the Software as a service (SaaS) layers of the Mobile Cloud Computing (MCC) paradigm, respectively. The users defined their policies to manage their privacy directly in PRECISE. This latter used (i) ontologies to shape the contextual and the user's information, (ii) semantic rules to define the privacy policies, and (iii) semantic reasoning to infer the decision whether or not request recommendations from a specific context-aware service. Although the users can hide their locations from the context-aware services, these data are collected and managed by a third-party, which is the PRECISE administrator in this work.

For their part, Funke et al. [Funke et al., 2015] proposed an end-to-end privacy architecture that facilitated the privacy at the edge paradigm and provided policy-driven control. The existing privacy enhancing technologies are applied and controlled according to the privacy policies. Several trust parties are required in this solution, such as the authenticity is included via an Identity Provider (IdP) as well as the authorization is decoupled from the IdP via pseudonyms issued by a trusted third-party.

Publish/subscribe system features allowed indirect, anonymous and multicast interactions among smart grid services. In this context, Duan et al. [Duan et al., 2016] proposed a Data-Centric Access Control Framework, called DCACF to support secure access control in a pub-

lish/subscribe model. According to the authors, this framework helped to build scalable smart grid services, while keeping features of service interactions and data confidentiality at the same time. The broker re-encrypted the event with the subscriber's public key, and the subscriber used its private key to decrypt the re-encrypted event. Thus, the publisher and the subscriber in DCACF did not need to share any key, reducing the key management burden of the publishers.

Henze et al. [Henze et al., 2016] proposed a User-driven Privacy Enforcement for Cloud-based Services, called UPECSI for the IoT domain. The proposed solution allowed users to enforce all their privacy requirements before any sensitive data are uploaded to the cloud. Moreover, UPECSI offered a transparent and adaptable interface for configuring the users' privacy requirements. A trusted third-party is used in order to certify each entity's identity. Moreover, UPECSI enabled developers of cloud services to integrate privacy functionality into the development process of cloud services. A trusted third-party (TTP) is used to audit the correct implementation of the cloud service.

Huertas Celdrán et al. [Huertas Celdrán et al., 2016] proposed a framework, called Semantic web-based Context Management (SeCoMan) that provided some support for developing context-aware smart applications that preserved the users' privacy. SeCoMan aimed at offering a set of predefined queries using a semantic oriented IoT vision. In fact, SecoMan employed an ontology to model the description of entities, reason over data to obtain useful knowledge, and define context-aware policies. SeCoMan defined policies using semantic rules. These policies helped the users to share their location to the right consumers, at the right granularity, at the right place, and at the right time. However, the privacy protection is fulfilled in a location-limited level. Moreover, SeCoMan can be considered as a trusted third-party that managed the users' privacy.

Da Costa et al. [Da Costa et al., 2017] proposed a privacy-aware Sensing as a Service (pSaaS) using a new privacy model. This model provided a Privacy Enforcement Point (PEP) that intermediated the connected data providers and the data consumer, implementing an in-network verification process. This process reasoned about inference intention and personal information in order to deny access or degrade data utility to specific parts of the IoT data stream. Virtual sensor outputs are defined by semantic representation and SPARQL is used to infer about privacy policy conditions defined using semantic annotation.

Drosatos et al. [Drosatos et al., 2017] proposed PrivTAM, a system for calculating privacy-preserving Television Audience Measurement (TAM) ratings through SmartTV technology. The core of PrivTAM was a privacy-preserving cryptographic protocol, which got as input the viewing records from the users' SmartTVs and outputted the TAMs by performing secure multi-party computations. In fact, SmartTVs communicated over the Internet to calculate aggregated measurements. SSL/TLS sockets are used to secure the communication between the different entities. However, PrivTAM required a central third-party, called TAM Aggregator that coordinated the TAM computation, verified the validity of the records, collected the encrypted results, and provided the compensation to the participants.

Wu et al. [Wu et al., 2017] proposed a lightweight and anonymous authentication scheme

for wearable devices with the help of cloud server. The proposed scheme reached mutual authentication while keeping the anonymity of the IoT devices. Only lightweight cryptographic operations including hash function and exclusive-or operations are employed. The proposed scheme can be divided into three phases: initialization, pairing, which enabled the smartphone and the wearable device to know the existence of each other, and authentication, which constructed the session key for the latter information transmission after pairing. However, the authors considered the cloud server as trustful thanks to its own security mechanism. Thus, the cloud server stored all the critical identity information about both the wearable device and the user's smartphone.

The main challenge with such solutions is that they required full trust in the third-party for the data protection and privacy preservation.
After providing a detailed review on privacy-preserving approaches in the IoT domain, we analyze them on the following section according to our defined analysis criteria.

1.4 Analysis

Both Table 1.3 and Table 1.4 categorize all the privacy-preserving approaches in the IoT domain introduced in Section 1.3. Five axes are used simultaneously to qualify the state-of-the-art, namely the privacy design strategies (Section 1.2.2), the three-layered privacy model (Section 1.2.3), the privacy-preserving architectures (Section 1.2.4), the privacy-preserving mechanisms (Section 1.2.5), and the IoT data lifecycle. We notate the lifecycle phases in the following tables as: registration (**R**, for short), collection (**C**, for short), transmission (**T**, for short), storage (**S**, for short), processing (**P**, for short), and end-to-end (**E2E**, for short).

Except for one [Funke et al., 2015], all the rest surveyed solutions are only concerned with one or two IoT data phases, and generally not address the whole IoT lifecycle. Few of these solutions addressed privacy at the collection time [Wong and Kim, 2014] [Birman et al., 2015] [Ukil et al., 2015]. Preserving privacy in the collection phase is essential and can affect the whole data lifecycle. Thus, privacy should be preserved before the transmission phase instead of trying to preserve it when the data are already stored in the consumer's data center. After being collected, the IoT data are sent to the joint or recipient spheres. Several approaches are proposed in order to protect (i) the IoT device anonymous authentication [Lai et al., 2014] [Wu et al., 2017], (ii) the data owner's anonymity [Birman et al., 2015] [Kravets et al., 2015], and (iii) the data confidentiality and security [Duan et al., 2016] during the transmission phase. Once transmitted, data should be stored to be available for analyzing. A high storage capacity is required to support the huge amount of data generated by IoT devices. In this context, several approaches [Huertas Celdrán et al., 2014] [Zyskind et al., 2015] [Henze et al., 2016] [Dorri et al., 2017b] relied on a centralized cloud to store the IoT data. While this requires some amount of trust in a third-party, it has some advantages in terms of scalability and ease of deployment. To overcome the centralized storage and the trust need, several distributed data storage systems [Hashemi et al., 2016] [Jayaraman et al., 2017] [Shafagh et al., 2017] appeared.

Table 1.3: List of the privacy-preserving approaches in the IoT domain

	Architecture	IoT Data Lifecycle	User Sphere				Joint Sphere		Recipient Sphere			
			Minimize	Hide	Separate	Aggregate	Secure Communication	Anonymous Communication	Inform	Control	Enforce	Demonstrate
Perturbation-based mechanisms												
CPAL [Lai et al., 2014]	Cen	T		✓				✓			✓	
[Wong and Kim, 2014]	Dis	C	✓	✓				✓				
[Birman et al., 2015]	Dis	C+T	✓	✓		✓	✓					
[Funke et al., 2015]	TTP	E2E	✓	✓		✓		✓			✓	
DPA [Ukil et al., 2015]	Cen	C				✓	✓					
[Wu et al., 2017]	TTP	T		✓				✓				
Restriction-based mechanisms												
Incognito [Kravets et al., 2015]	Cen	T		✓				✓			✓	
DCACF [Duan et al., 2016]	TTP	T+P		✓		✓	✓			✓	✓	
UPECSI [Henze et al., 2016]	TTP	S					✓		✓	✓	✓	✓
[Drosatos et al., 2017]	TTP	P		✓		✓	✓					
[Jayaraman et al., 2017]	Dis	S+P		✓	✓	✓	✓				✓	
Aggregation-based mechanisms												
PAgIoT [González-Manzano et al., 2016]	Cen	P		✓	✓	✓		✓				
LPDA [Lu et al., 2017]	Cen	P	✓		✓	✓	✓					
[Guan et al., 2018]	Cen	P	✓	✓	✓	✓		✓				✓
[Abdallah and Shen, 2018]	Dis	P	✓		✓	✓	✓					
3PDA [Liu et al., 2018]	Dis	P	✓		✓	✓	✓					
[Tonyali et al., 2018]	Dis	P	✓		✓	✓	✓					
[Wang et al., 2018]	Dis	P	✓	✓	✓	✓		✓				✓
APPA [Guan et al., 2019]	Cen	P	✓		✓	✓	✓					
Semantic-based mechanisms												
PRECISE [Huertas Celdrán et al., 2014]	TTP	S+P	✓	✓				✓		✓	✓	
[Hosseinzadeh et al., 2016]	Cen	S+P					✓			✓	✓	✓
SeCoMan [Huertas Celdrán et al., 2016]	TTP	P	✓	✓			✓				✓	
ORDM [Wang et al., 2016]	Cen	P					✓				✓	
[Da Costa et al., 2017]	TTP	P					✓				✓	

Table 1.4: List of the privacy-preserving approaches in the IoT domain (continued)

	Architecture	IoT Data Lifecycle	User Sphere				Joint Sphere		Recipient Sphere			
			Minimize	Hide	Separate	Aggregate	Secure Communication	Anonymous Communication	Inform	Control	Enforce	Demonstrate
Blockchain-based mechanisms												
[Zyskind et al., 2015]	Dis	S+P		✓		✓	✓	✓	✓	✓	✓	✓
ChainAnchor [Hardjono and Smith, 2016]	Dis	R		✓			✓	✓		✓		
[Hashemi et al., 2016]	Dis	S+P		✓	✓			✓		✓		
FairAccess [Ouaddah et al., 2016]	Dis	P		✓				✓	✓	✓	✓	✓
[Dorri et al., 2017b]	Dis	S+P		✓			✓	✓	✓	✓	✓	✓
[Shafagh et al., 2017]	Dis	S		✓	✓			✓	✓	✓	✓	✓
Ancile [Dagher et al., 2018]	Dis	S+P		✓			✓	✓	✓	✓	✓	✓

Finally, privacy at processing time is addressed by exposing the raw IoT data to the data consumer or only exposing the calculated result obtained after the data producers collaboration. The first solution required the definition of privacy policy and the use of technical protection mechanism, namely access control to prevent any privacy policy violations[Hosseinzadeh et al., 2016] [Huertas Celdrán et al., 2016] [Ouaddah et al., 2016] [Wang et al., 2016] [Da Costa et al., 2017] [Dagher et al., 2018]. For the second solution, lightweight aggregation protocols [González-Manzano et al., 2016] [Lu et al., 2017] [Abdallah and Shen, 2018] [Liu et al., 2018] [Guan et al., 2019] and multi-party computations [Drosatos et al., 2017] [Tonyali et al., 2018] are used in the literature.

Only the architecture proposed by Funke et al. [Funke et al., 2015] allowed E2E privacy in IoT in the sense that privacy components can reside in the device domain as well as in the cloud domain, controlled by a policy. However, this work did not consider the totality of the privacy design strategies that guaranteed the data owner's legal rights.

Moreover, Tables 1.3 and 1.4 show that the existing approaches address a few privacy design strategies while omitting others. Applying both perturbation-based and aggregation-based mechanisms on the data before they become out of the data owner's control enhance the compliance with the first four privacy design strategies by the perturbation-based approaches. Furthermore, both the restriction-based and the semantic-based mechanisms enhanced the data owner's right to make in place a legal requirement compatible privacy policy over the data and enforce it on the recipient sphere. Thus, the data owner can define a set of privacy preferences and the data consumer needs to enforce them. However, demonstrating compliance with the defined

privacy policy is not straightforward. More recently, the blockchain has been proposed as a possible solution for managing and enforcing privacy-preserving needs. As shown on Table 1.4, most of the blockchain-based approaches reach the compliance with the last four privacy design strategies namely, *inform*, *control*, *enforce*, and *demonstrate*. Moreover, the ability to create smart contracts make blockchain suitable for IoT, where strict regulations govern how sensitive data can be used. Information exchange using smart contracts is transparent, conflict-free, and eliminated the need for a trust third-party as the blockchain executed the data sharing based on the conditions of the contract [Dagher et al., 2018].

1.5 Summary

Privacy in the IoT domain had already been reviewed in several surveys. While [Fernández-Alemán et al., 2013] [Ziegeldorf et al., 2014] [Seliem et al., 2018] [Sen et al., 2018] focused on analyzing the challenges and threats of IoT in the context of entities and information flows. However, these surveys do not address the legal privacy principle coverage according to the three areas namely, the user, the joint and the recipient spheres. Moreover, previous surveys focused on some data phases and not the whole IoT data lifecycle. In this chapter, we surveyed the latest approaches about privacy in several domains, and especially in the IoT domain. First, we gave an overview of the privacy notion and we introduced some analysis criteria, which are used to evaluate the surveyed papers. This evaluation is analyzed and shown by the three tables (Table 1.2, Table 1.3, and Table 1.4).

As seen in this survey, a prominent issue is the lack of an end-to-end solution for privacy in the IoT domain that covers the legal privacy principles represented by the privacy design strategies. In the next chapter, we introduce our contribution, which is a European Legal compliant ontology for supporting preserving IoT PrivacY, called LIoPY in order to address the IoT privacy issue.

Chapter 2

LIoPY: European Legal compliant ontology for supporting preserving IoT PrivacY

Table of Contents

2.1 Introduction . . . 35
2.2 Existing IoT and privacy ontologies . . . 36
2.3 Overview of the European Legal compliant ontology for supporting preserving IoT PrivacY (LIoPY) . . . 41
2.3.1 Building LIoPY . . . 41
2.3.2 LIoPY's Modules . . . 50
2.4 Privacy Preferences Model . . . 52
2.4.1 Collected-data-centric privacy-preserving perspective . . . 53
2.4.2 Shared-data-centric privacy-preserving perspective . . . 54
2.5 Summary . . . 69

2.1 Introduction

In an IoT environment, smart devices collaborate among one another and with other physical and virtual objects in order to perform high-level tasks. Users surrounded by several ambient sensors may be unaware of the use and the sharing of their private information, which can create great concerns for privacy [Chen et al., 2004]. In a such open and dynamic environment, ontology is a good manner to describe the generated data and enable the users to create fine-grained privacy preferences for their collected IoT data. Indeed, semantic enables the implementation issue abstract. It also enables identifying and defining the basic concepts for the privacy aspect description in a domain-independent manner. Finally, ontology use enables to resolve the inter-

operability issue on different concepts used by IoT actors and resources across heterogeneous domains to define the shared data. Therefore, we survey in Section 2.2 some existing ontologies related to the IoT domain and the privacy issue that we consider relevant to our research problem. In Section 2.3, we overview our proposed European Legal compliant ontology for supporting preserving IoT PrivacY, called LIoPY that provides a solution for overcoming the existing ontology-based model limitations. After that, we present our defined privacy preferences model in Section 2.4 before summarizing the content of this chapter in Section 2.5.

2.2 Existing IoT and privacy ontologies

With the popularity and power of semantics, researchers have used ontologies in order to preserve privacy in the IoT environment.

Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) [Chen et al., 2004] is proposed in 2004. SOUPA combined many vocabularies from different consensus ontologies in order to assist the ubiquitous and pervasive applications' developers to build ontology-driven applications. Figure 2.1 depicts an overview of this ontology that included concepts such as Agent with associated properties, such as Believes, Desires, Intends, Event, Time, Space, and Policy for security and privacy. In SOUPA, both computational entities and human users can be considered as agents. Moreover, a policy is specified by a user or a computing entity to restrict the personal information type that can be shared by the public services. However, these restrictions are limited to the actor that performed the action, the recipient that received the effect after the action execution, and the location/time at where/which the action is performed.

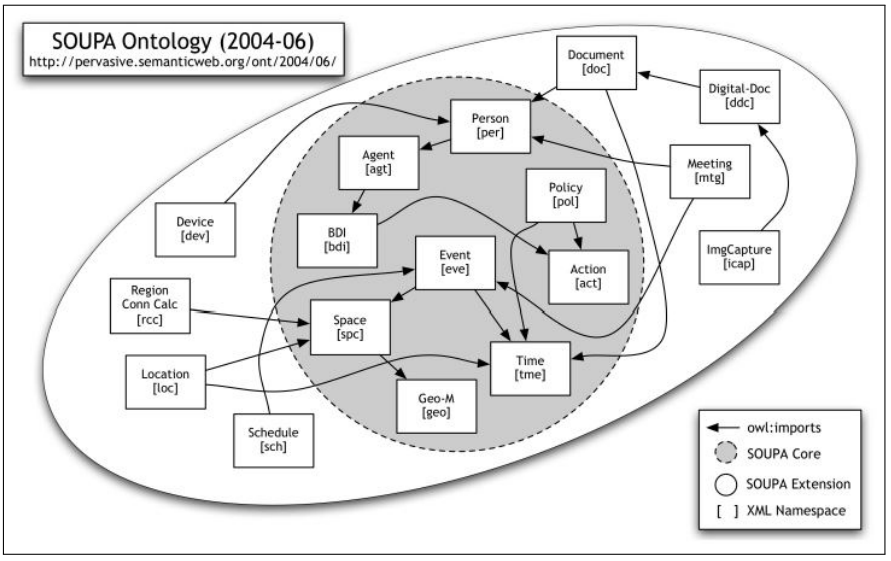


Figure 2.1: An overview of SOUPA [Chen et al., 2004]

Semantic Sensor Network (SSN) ontology [Haller et al., 2018] is published in 2009, updated and recommended by the World Wide Web Consortium (W3C) in 2017. Moreover, SSN imported a lightweight but self-contained core module, called SOSA (Sensor, Observation, Sample, and Actuator). SSN described sensors, actuators, samplers, and observations that are data collected through these sensors. Figure 2.2 provides an overview of the classes and properties in the SSN ontology core. For instance, SOSA recognized sosa:Sensor (e.g., a smart meter) that made sosa:Observation about some sosa:ObservableProperty (e.g., electric consumption) of a sosa:FeatureOfInterest (e.g., kitchen in a smart home). Although the SSN ontology modeled the sensor network domain, it lacked the relevant definitions to preserve privacy. Moreover, it did not provide reasoning provided by semantic technologies.

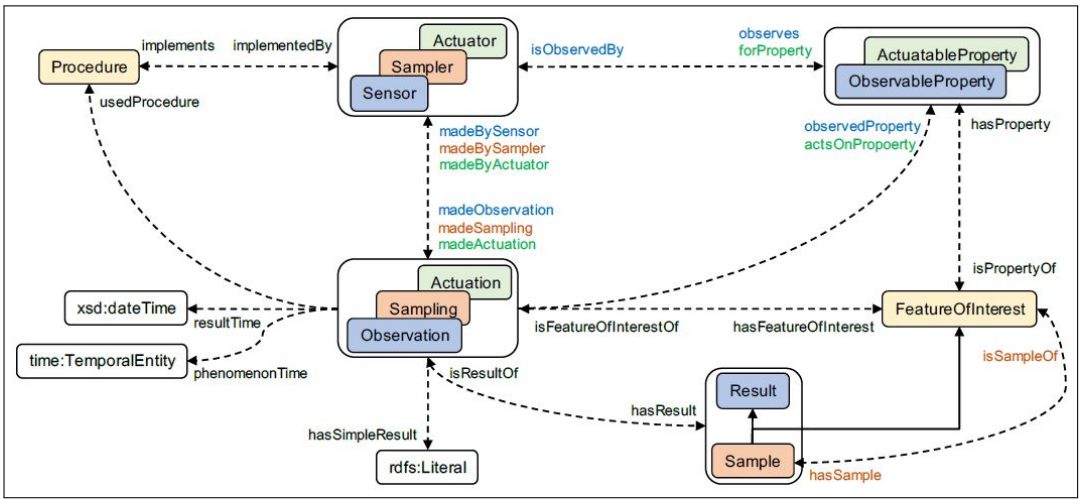


Figure 2.2: An overview of SSN ontology [Haller et al., 2018]

IoT-Lite ontology [Bermudez-Edo et al., 2017] is proposed as an instantiation of the semantic sensor network (SSN) ontology to describe key IoT concepts. It allowed interoperability and discovery of sensed data in heterogeneous IoT platforms by a lightweight semantics. IoT-Lite's intent is not to be a full ontology for the IoT, but to be a lightweight ontology that allowed fast annotation and processing time. Figure 2.3 depicts an overview of the IoT-Lite's information model, which included three well-accepted items in the classification of IoT entities, namely Entities or objects; resources or devices; and services, called iot-lite:Entity, ssn:Device, and iot-lite:Service, respectively. IoT-Lite ontology focused on reducing the processing time of a query time response. However, IoT-Lite ontology did not take into account the IoT privacy issue.

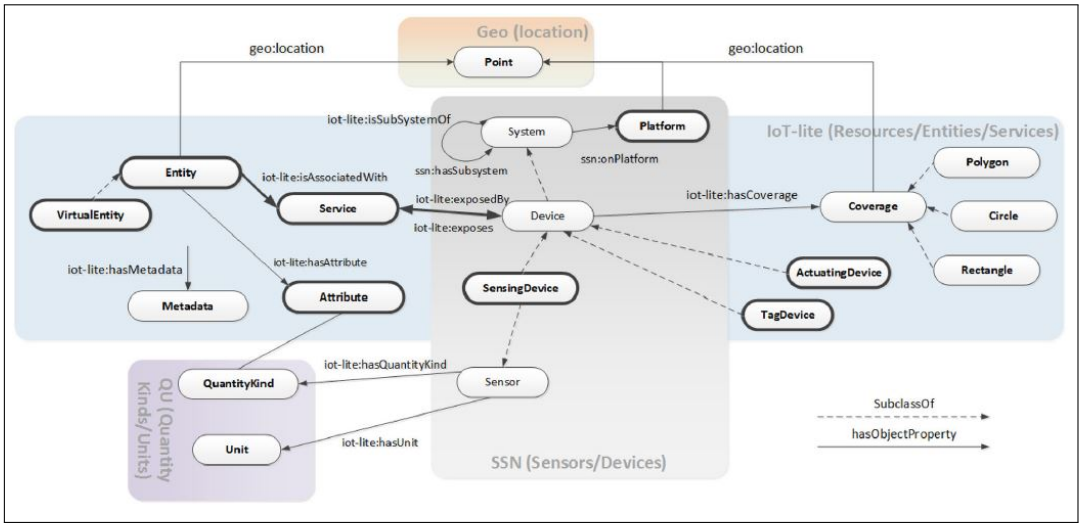


Figure 2.3: An overview of IoT-Lite ontology [Bermudez-Edo et al., 2017]

Privacy Preference Ontology (PPO) [Sacco and Passant, 2011] is a lightweight vocabulary for Linked Data on top of the existing Web Access Control. PPO enabled users to create fine-grained privacy preferences for their data. The vocabulary is designed to describe the access privilege to the data and restrict data access. Figure 2.4 depicts some classes and properties of the privacy preference ontology. The main class ppo:PrivacyPreference defined a privacy preference that is linked with some properties, which defined (i) which statement, resource and/or graph is to be restricted, (ii) which access privilege should be granted, and (iii) which attribute patterns a requester must satisfy. The PPO is essentially used to provide fine-grained privacy preferences for RDF data and no other data type.

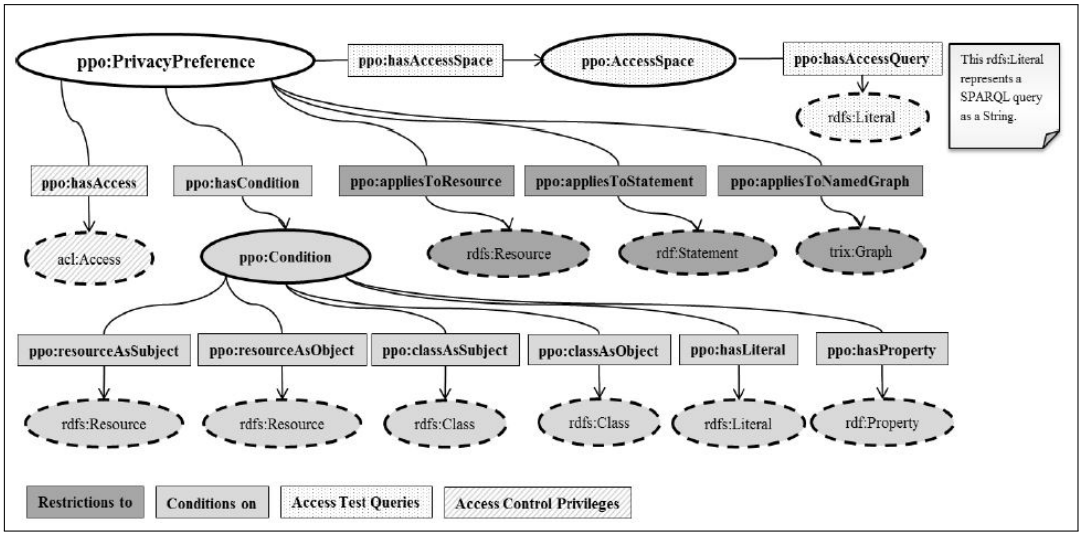


Figure 2.4: An overview of PPO [Sacco and Passant, 2011]

Data Privacy Ontology (DPO) [Bawany and Shaikh, 2017] is proposed in order to preserve privacy for ubiquitous computing. Figure 2.5 depicts an overview of the data privacy ontology. The principal class is IPEntity with three subclasses, namely DataHolder, Consumer, and Data that has a relationship with the PrivacyPolicy class. Although the data access location and time are considered in DPO, the privacy requirements like consent, purpose, and disclosure are neglected in this ontology. Moreover, DPO modeled privacy policy terms, however it did not consider the description of the IoT concepts, such as device, sensor, or observation.

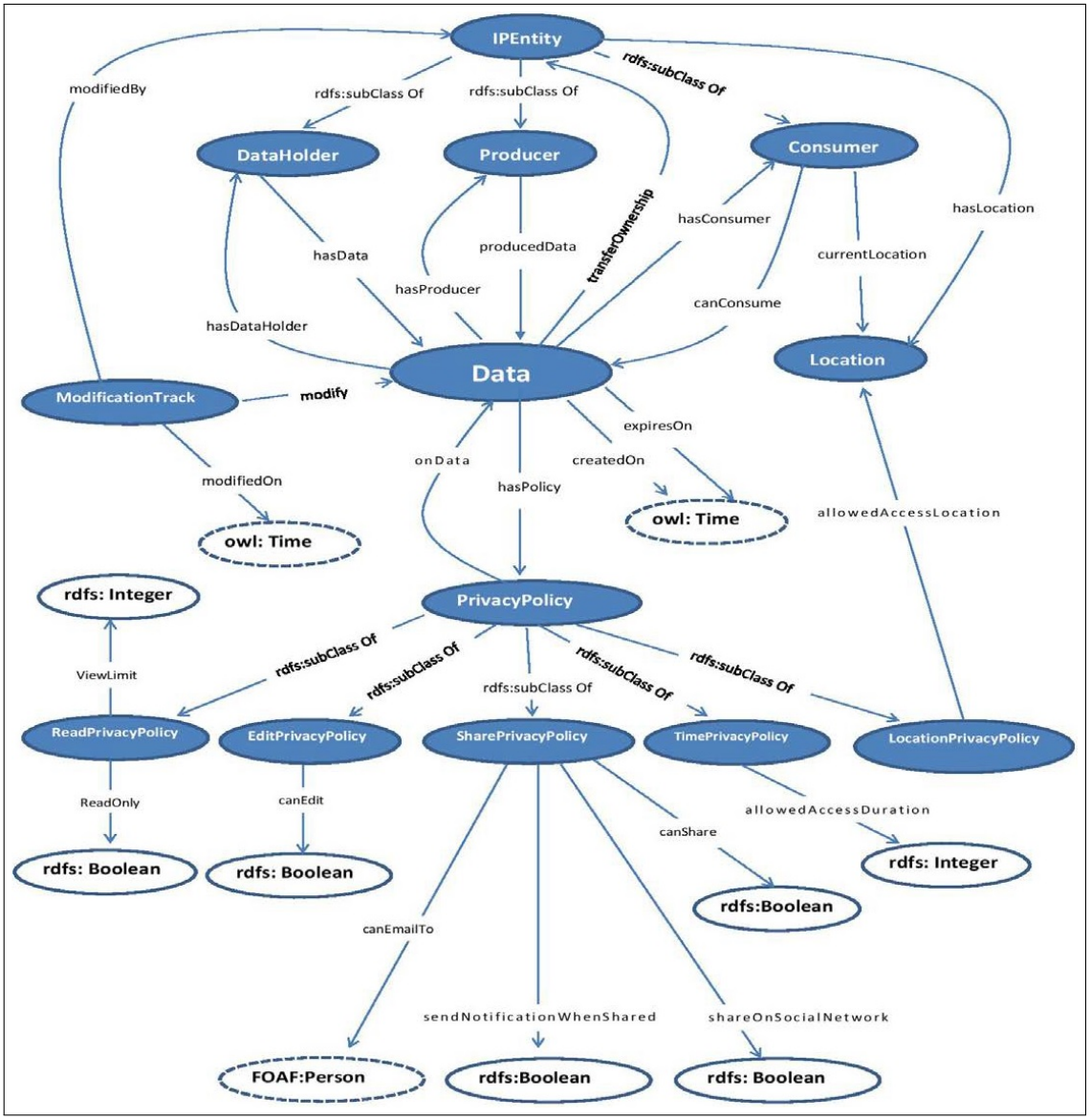


Figure 2.5: An overview of DPO [Bawany and Shaikh, 2017]

Ontology-based Resource Description Model (ORDM) [Wang et al., 2016] is proposed in order to describe resources in the IoT environment, which are described by the attribute, state, control, historical information and privacy classes. Figure 2.6 depicts an overview of the ontology-based resource description model. The Attribute class defined the inherent information of the device, such as the device type, model, and range of the sensed values. The data description is made

in the State class, which provided the current data captured by the sensor with their associated data unit. ORDM focused on protecting the device from illegal access or control thanks to the Privacy class definition. However, no fine-grained access control to the produced data is offered by ORDM. Indeed, the users that can access the resource are fixed in the proposed ontology without any reasoning or clear criteria.

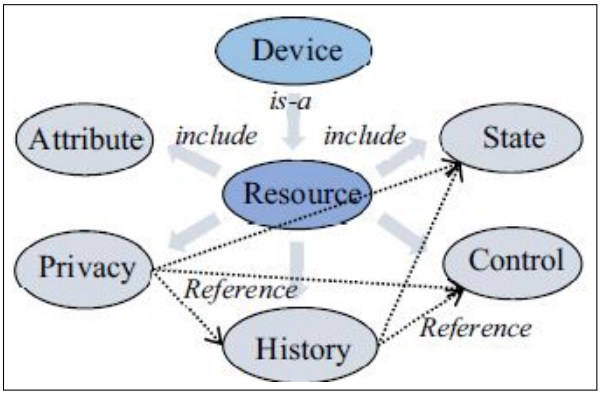


Figure 2.6: An overview of ORDM [Wang et al., 2016]

Table 2.1 presents a comparison between the surveyed ontologies according to the IoT description, supporting IoT privacy and addressing the privacy requirements. Table 2.1 shows that none of the existing ontologies fully describes the IoT device and covers all the privacy requirements proposed by the European legislation [Directive 95/46/EC, 1995] [GDPR, 2016] and privacy standards [OECD, 1981] [ISO/IEC29100, 2011] while dealing with the IoT privacy issue.

Table 2.1: Existing IoT ontologies comparison

	IoT Description	**Privacy Support**	**Privacy Requirements**					
			Consent	Purpose	Retention	Operation	Condition	Disclosure
SOUPA [Chen et al., 2004]	+	+	+	-	-	+	+	-
ORDM [Wang et al., 2016]	+	+	-	-	-	-	+	-
PPO [Sacco and Passant, 2011]	-	+	+	-	-	+	+	-
DPO [Bawany and Shaikh, 2017]	-	+	-	-	+	+	+	-
SSN [Haller et al., 2018]	+	-	-	-	-	-	-	-
IoT-Lite [Bermudez-Edo et al., 2017]	+	-	-	-	-	-	-	-

In summary, none of the existing ontologies are fully providing a solution to data management for IoT privacy since they are only covering a part of the IoT domain description while omitting fine-granularity privacy management for each IoT device output. Furthermore, no combination of the existing ontologies is able to provide a complete level of reasoning over the privacy requirements proposed by privacy laws and standards. Thus, we propose a European

Legal compliant ontology for supporting preserving IoT PrivacY, called LIoPY that provides a solution for overcoming the aforementioned existing ontology-based model limitations.

2.3 Overview of the European Legal compliant ontology for supporting preserving IoT PrivacY (LIoPY)

In this section, we provide an overview of LIoPY by first, detailing its process of building that followed a well-known ontology design methodology, then providing the considered design principles. After that, we present LIoPY's modules and its main classes and properties.

2.3.1 Building LIoPY

The process of building LIoPY followed the MethOntology methodology [Fernández-López et al., 1997], which is one of the most famous ontology design methodologies [Cristani and Cuel, 2005]. It defines seven activities that an ontology's developer needs to carry out when building an ontology:

1. **Specification phase**: its goal is to produce either an informal, semi-formal or formal ontology specification document written in natural language and included the ontology purpose, the formality level of the implemented ontology, and the ontology scope (i.e., a set of represented terms in the ontology) [Fernández-López et al., 1997].

   LIoPY is proposed for filling the gap between the IoT ontologies (like IoT-lite and SSN) that do not support a semantic representation of the privacy requirements and the privacy laws that cannot be easily integrated into real-world applications. Besides the semantic representation of the IoT domain, LIoPY aims at identifying and defining the basic concepts for the description of privacy requirements to enable its integration into reasoning frameworks. LIoPY can be used by both IoT data owner and IoT data consumer in order to preserve the privacy of the shared data between these two actors. In fact, LIoPY enables the IoT data owner to define some privacy preferences, enables the IoT data consumer to define its privacy terms of service, and matches the data owner's preferences and the consumer's terms of service in order to generate a common privacy policy that can be applied to preserve the data owner privacy in the IoT environment while handling the data by the consumer. Such matching cannot be possible if the data owner's preferences and the consumer's terms of service are not expressed by the use of the same privacy vocabulary that describes the privacy requirements.
   Table 2.2 shows the LIoPY's specification document that is represented by using a natural-language semi-formal format based on a middle-out approach, which consists in gathering a set of terms that must be included in the ontology whether or not the ontology designer knows their meaning at this stage of the ontology development process and classify them on a concept classification tree [Fernández-López et al., 1997]. The reason behind the use of the middle-out approach is that it enables identifying the ontology primary concepts that

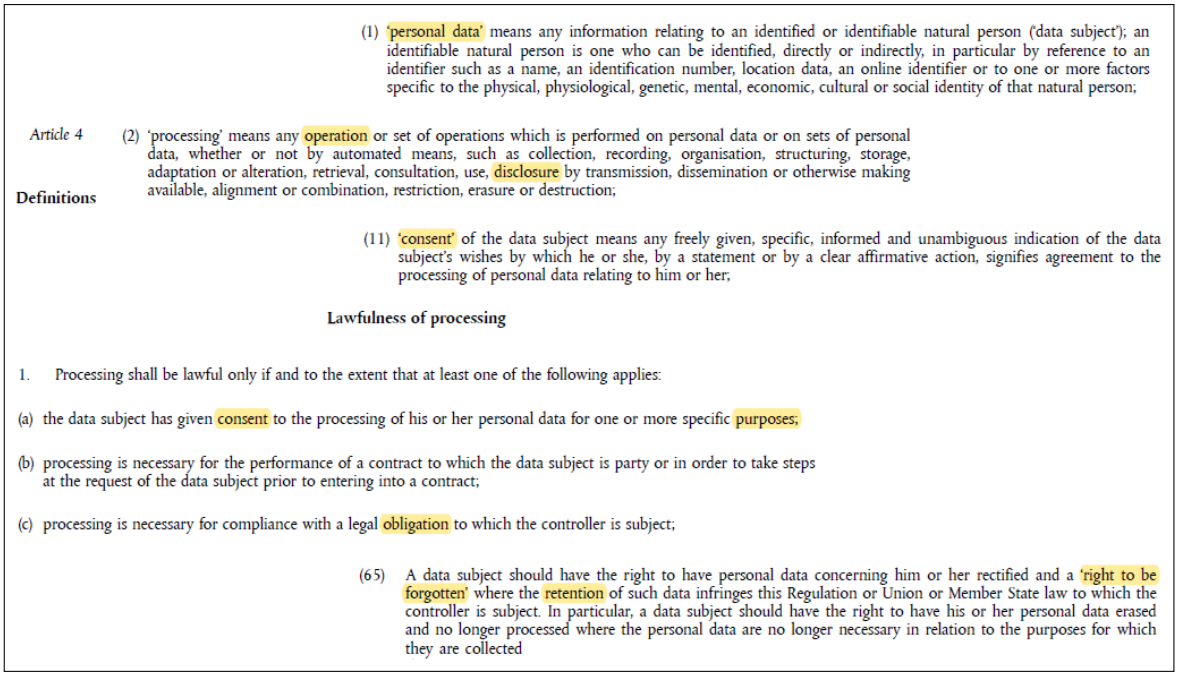
can be specialized or generalized after reaching agreement on their definition, only if they are necessary. As a result, the used terms are more stable, and less re-work and overall effort are required [Fernández-López et al., 1997]. The scope's granularity level is classified as *high* thanks to the rich set of terminologies represented in the ontology.

Table 2.2: LIoPY's specification document

ONTOLOGY REQUIREMENT SPECIFICATION DOCUMENT
Domain: Privacy preferences and requirements in the IoT domain **Date:** September, 1^{st} 2017 **Conceptualized by:** Faiza LOUKIL **Implemented by:** Faiza LOUKIL **Purpose:** Ontology about incorporate privacy legislation into privacy policies while considering several privacy requirements in the IoT domain. It enables the data owners to create fine-grained privacy preferences for their data. Moreover, it aims at making the IoT devices more autonomous by giving them the capability to infer the consumer's access rights according to the owner's privacy preferences. This ontology can be used in several IoT domains, namely smart home, smart city, and healthcare. **Level of Formality:** Semi-formal. **Scope:** • **List of privacy preferences and requirements:** consent, purpose, disclosure, retention, operation, condition, etc. • **List of concepts:** owner, consumer, producer, role, decision, iot-device, system, sensor, actuator, sampler, observation, actuation, sampling, default-value, device-capability, device-feature, device-feature-of-interest, device-input, device-output, data-category, data-quality, privacy-rule, privacy-obligation, terms-of-service, privacy-policy, privacy-obligation, etc. • **List of properties:** owns, has-consent, has-role, has-decision, has-capability, has-feature, made-observation, made-actuation, made-sampling, iot-device-property, capability-connection-technology, capability-power-consumption, actor-role, sensor-observation, min-default-value, max-default-value, observation-feature-of-interest, observation-quality, observation-category, data-category-rule, terms-of-service-policy, etc. **Sources of knowledge:** • SOUPA [Chen et al., 2004] • SSN ontology [Haller et al., 2018] • IoT-Lite ontology [Bermudez-Edo et al., 2017] • OECD privacy guidelines [OECD, 1981] • Directive 95/46/EC [Directive 95/46/EC, 1995] • General Data Protection Regulation (GDPR) [GDPR, 2016] • ISO 29100 privacy framework [ISO/IEC29100, 2011]

2. **Knowledge Acquisition phase**: its goal is to search and list knowledge sources through non-structured interviews with experts, informal and formal text analysis, and structured interviews with experts to get detailed knowledge about concepts, their properties, and their relationships [Fernández-López et al., 1997].

   LIoPY is based on several sources of knowledge, such as (i) existing W3C recommended ontologies, such as SOUPA ontology [Chen et al., 2004], SSN ontology [Haller et al., 2018], and IoT-Lite ontology [Bermudez-Edo et al., 2017], (ii) privacy guidelines [OECD, 1981] and European privacy laws [Directive 95/46/EC, 1995] [GDPR, 2016], and (iii) ISO 29100 privacy framework [ISO/IEC29100, 2011]. By inspecting similar IoT ontologies, we elaborate a first glossary with terms potentially relevant. This first list of terms is refined by contrasting it against figures that are given in some books and scientific reports in order to include or remove terms in the glossary. Moreover, we elaborate an informal text analysis to study the main concepts related to the privacy requirements given in privacy law texts and related guide reports. Figure 2.7 depicts some extracts from the GDPR document [GDPR, 2016] that is one of the analyzed knowledge sources in this phase. While reading the document, we analyze the text and highlight the main concepts that are related to the ontology domain.



(1) 'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;

Article 4

Definitions

(2) 'processing' means any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction;

(11) 'consent' of the data subject means any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her;

Lawfulness of processing

1. Processing shall be lawful only if and to the extent that at least one of the following applies:

(a) the data subject has given consent to the processing of his or her personal data for one or more specific purposes;

(b) processing is necessary for the performance of a contract to which the data subject is party or in order to take steps at the request of the data subject prior to entering into a contract;

(c) processing is necessary for compliance with a legal obligation to which the controller is subject;

(65) A data subject should have the right to have personal data concerning him or her rectified and a 'right to be forgotten' where the retention of such data infringes this Regulation or Union or Member State law to which the controller is subject. In particular, a data subject should have the right to have his or her personal data erased and no longer processed where the personal data are no longer necessary in relation to the purposes for which they are collected

Figure 2.7: Informal text analysis of the GDPR [GDPR, 2016]

After extracting the main concepts, a formal text analysis contributes to identify the structure and the kind of knowledge of each concept, such as concepts, attributes, or relationships. Table 2.3 shows the LIoPY's knowledge acquisition document that defines for each potentially relevant term its source, name, and kind.

Table 2.3: LIoPY's knowledge acquisition document

ONTOLOGY KNOWLEDGE ACQUISITION DOCUMENT				
Terms' Source	**Terms**	**Term's Kind**		
		Concept	**Attribute**	**Relationship**
SSN/SOSA ontology	System	✓		
	Actuator	✓		
	Sampler	✓		
	Sensor	✓		
	Actuation	✓		
	Sampling	✓		
	Observation	✓		
	madeActuation			✓
	madeSampling			✓
	madeObservation			✓
GDPR, OECD, ISO 29100	Condition	✓		
	Consent	✓		
	Disclosure	✓		
	Operation	✓		
	Purpose	✓		
	Retention	✓		
	Decision	✓		
	Data category	✓		
	Privacy rule	✓		
	Terms of service	✓		
	Privacy policy	✓		
	Privacy obligation	✓		
	Producer	✓		
	Owner	✓		
	Consumer	✓		
	Role	✓		
	retention-duration		✓	
	explicit-identifier-category		✓	
	sensitive-category		✓	
	quasi-identifier-category		✓	
	non-sensitive-category		✓	
	owns			✓
	has-consent			✓
	has-decision			✓
	has-role			✓
	has-category			✓
	has-rule			✓
Scientific reports	Default value	✓		
	Device input	✓		
	Device output	✓		
	Device property	✓		
	Capability	✓		
	Feature	✓		
	capability-power-consumption		✓	
	has-device-input			✓
	has-device-output			✓
	has-property			✓

3. **Conceptualization phase**: its goal is to structure the domain knowledge in a conceptual model that describes the problem and its solution in terms of the domain vocabulary identified in the ontology specification phase [Fernández-López et al., 1997].

   LIoPY's conceptualization is split into two phases. The first step is covered by the specification and the knowledge acquisition phases, where the most of the terms are collected in order to produce a complete glossary of terms including concepts, instances, verbs, and properties. The second step consists in providing more details about each concept by defining its properties (both object and data). LIoPY's conceptual model document includes the concept classification trees, verb diagrams, and the data property dictionary. Figure 2.8 shows LIoPY's concept classification trees and verb diagrams generated after the conceptualization phase. White rectangles depict the imported concepts preceded by their source ontology's acronym and blue rectangles depict the newly defined concepts in LIoPY.

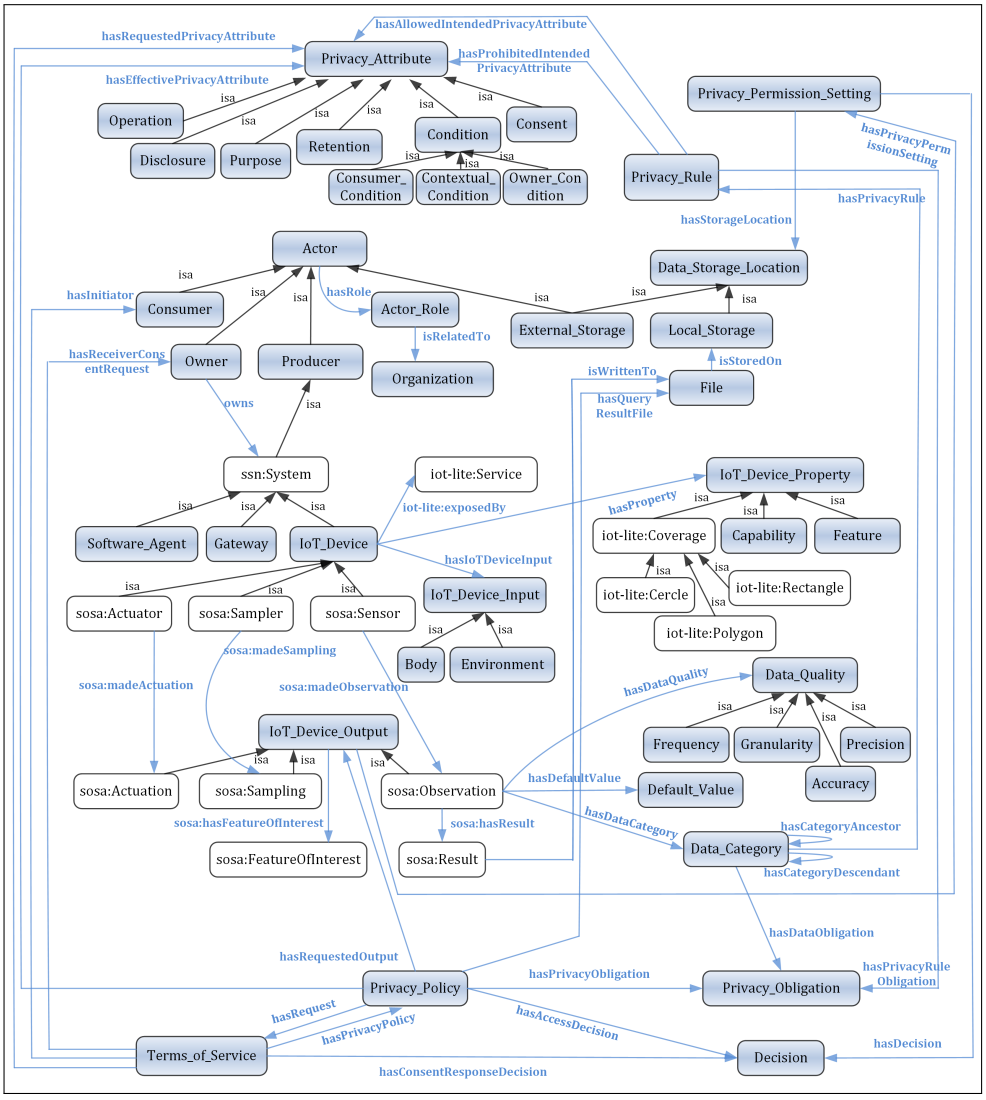


Figure 2.8: LIoPY's conceptual model document: concepts and relationships

Moreover, Table 2.4 shows the LIoPY's data property dictionary that defines for each con-

cept its data properties and meaning in a declarative way.

Table 2.4: LIoPY's conceptual model document: data property dictionary

ONTOLOGY CONCEPTUAL MODEL DOCUMENT: data property dictionary		
Concept	**Concept's data properties**	**Data property's meaning**
Owner	owner_birthday owner_country	owner's date of birthday owner's country name
Frequency	frequency_unit frequency_value	device's frequency unit device's frequency value
Granularity	granularity_level	device's granularity level
Accuracy	accuracy_level	device's accuracy level
Precision	precision_unit precision_value	device's precision unit device's precision value
Default_Value	min_default_value max_default_value	default value's minimum default value's maximum
File	file_type storage_path	type of a file storage path of a file
Body	body_height body_weight	height of a body weight of a body
sosa:Observation	observation_name observation_description	name of an observation observation's description
IoT_Device_Property	on_state property_description	device's state device's description
Capability	capability_connection_technology capability_power_consumption capability_valid_period capability_latency	device's connection technology device's power consumption device's valid period device's latency
Feature	device_manufacturer device_name device_serial_number	device's manufacturer device's name device's serial number
Retention	retention_duration_per_day	retention's duration
Privacy_Permission_Setting	operation_beginning_time operation_frequency operation_ending_time	device's permission to start the operation operation frequency device's permission to finish the operation
Privacy_Policy	take_effect_date	policy's beginning date
Terms_of_Service	requested_data_name requested_collect_beginning_date requested_collect_ending_date	device's output name device's output collect beginning time device's output collect ending time
Result	result_name result_value result_unit sosa:resultTime	device output's simple result name device output's simple result value device output's simple result unit device output's simple result collection time

More details about the modeled LIoPY's concepts are provided in the next sections.

4. **Integration phase**: its goal is to reuse the definitions already built into other ontologies instead of starting from scratch [Fernández-López et al., 1997].

   In order to be as interoperable as possible, LIoPY imports both SSN ontology [Haller et al., 2018] and IoT-lite ontology [Bermudez-Edo et al., 2017] to point out to some classes and extend it with appropriate privacy and security properties. Although SSN and IoT-lite ontologies present knowledge in the domain of sensor networks, they lack the relevant definitions of privacy requirements. The integration phase aims at producing an integration document that enumerates the reused concepts. Table 2.5 shows the LIoPY's integration document that defines for each reused term, its name in the conceptual model, its name in the reused ontology, and the reused ontology's acronym.

Table 2.5: LIoPY's integration document

ONTOLOGY INTEGRATION DOCUMENT		
Term in LIoPY's conceptualization and LIoPY's implementation	**Term's name in the reused ontology**	**Reused ontology's acronym**
ssn:System	System	SSN
sosa:Actuator	Actuator	SSN/SOSA
sosa:madeActuation	madeActuation	SSN/SOSA
sosa:Actuation	Actuation	SSN/SOSA
sosa:Sampler	Sampler	SSN/SOSA
sosa:madeSampling	madeSampling	SSN/SOSA
sosa:Sampling	Sampling	SSN/SOSA
sosa:Sensor	Sensor	SSN/SOSA
sosa:madeObservation	madeObservation	SSN/SOSA
sosa:Observation	Observation	SSN/SOSA
sosa:hasResult	hasResult	SSN/SOSA
sosa:Result	Result	SSN/SOSA
sosa:resultTime	resultTime	SSN/SOSA
sosa:FeatureOfInterest	FeatureOfInterest	SSN/SOSA
iot-lite:exposedBy	exposedBy	IoT-Lite
iot-lite:Service	Service	IoT-Lite
iot-lite:Coverage	Coverage	IoT-Lite
iot-lite:Cercle	Cercle	IoT-Lite
iot-lite:Polygon	Polygon	IoT-Lite
iot-lite:Rectangle	Rectangle	IoT-Lite

5. **Implementation phase**: its goal is to codify the ontology in a formal language in order to create a computable ontology [Fernández-López et al., 1997].

   LIoPY's implementation is provided in two programming languages. OWL/RDF provides a formal representation enabling the check of inconsistencies, the visualization of the ontology structure, and the ease of maintenance. Besides, the Semantic Web Rule Language (SWRL) provides easy support for the integration of inference over LIoPY's terms. In fact, some relationships cannot be expressed in OWL. For this, SWRL is used to define inference

rules that enable the generation of new knowledge based on the existing LIoPY's terms. The two versions are always synchronized. The implementation phase aims at producing an owl document that defines all the LIoPY's terms and inference rules. LIoPY's implementation document is an OWL file available online [1].

6. **Evaluation phase**: its goal is to carry out a technical judgment of the ontologies, their software environment, and documentation with respect to the defined specification document during each ontology building phase [Fernández-López et al., 1997].

   LIoPY is verified by using the verification OntoMetrics platform proposed in [Lantow, 2016]. OntoMetrics platform offers a web-interface to upload the OWL file, calculates a set of ontology quality metrics, and creates an XML-download file of calculated ontology quality metrics. The set of ontology quality metrics calculated for LIoPY is provided in the appendix (see Appendix A).

   In our evaluation, we aim at considering three aspects, such as (i) the domain scope to know *how well dœs the ontology represent the real world?*, (ii) the conceptual scope to know *what is the quality of the ontology in analogy to internal software quality characteristics?*, and (iii) the application scope to know *how well dœs the ontology in use as a component of an ontology-based information system?*. In order to answer these questions, we rely on the proposed ontology quality metrics provided by the OntoMetrics platform [Lantow, 2016]. Thus, we use the OntoMetrics platform in order to evaluate the ontology quality criteria, namely *accuracy*, *cohesion*, and *conciseness* in the domain scope, *computational efficiency* in the conceptual scope, and *efficiency* and *accuracy* in the application scope.

   The evaluation phase's output for this activity is an evaluation document that describes LIoPY's calculated ontology quality metrics. LIoPY's evaluation document is detailed in Table 2.6. A positive correlation is between the quality metrics and the quality criteria, including the accuracy, cohesion, and conciseness. Thus, LIoPY's quality increases with the increase of the computed metrics. We observe that only the quality metric named "Class richness" is reduced. This metric is related to the number of classes that have instances compared with the total number of classes. This reduced value can be explained by the evaluated LIoPY's version is not populated yet. Indeed, such user will have an instantiated LIoPY according to the defined privacy preferences. Indeed, in this version, only few classes are instantiated including Privacy_Attribute class and its sub-classes. A negative correlation is between the tangledness quality metric and the computational efficiency quality criteria. Thus, LIoPY's has a high computational efficiency only when the tangledness is very low, which is our case.

[1] https://drive.google.com/file/d/1NC9xYu2IT1m68G2NZr1agSwCC3qNvDsH/view?usp=sharing

Table 2.6: LIoPY's evaluation document using Quality Metrics and Quality Criteria Matrix

ONTOLOGY EVALUATION DOCUMENT				
Quality Metric	**Ontology Quality Criteria**			
	Accuracy	**Cohesion**	**Computational Efficiency**	**Conciseness**
Schema Metrics				
Attribute richness	0,6			
Inheritance richness	0,7			
Relationship richness	0,5			
Graph Metrics				
Absolute root cardinality		23		
Absolute leaf cardinality		49		
Average depth	2			
Maximal depth	5			
Average breadth	3,9			
Maximal breadth	23			
Tangledness			0,06	
Knowledgebase Metrics				
Average population				0,9
Class richness				0,2
Class Metrics				
Class inheritance richness	376,9			
Class children count				68

7. **Documentation phase**: its goal is to provide a documentation after each phase of the whole ontology development process [Fernández-López et al., 1997].

   LIoPY is documented since it provides a document after each building ontology phase. During the LIoPY development process, we produce a set of documents are produced, namely specification document, knowledge acquisition document, conceptual model document, integration document, implementation document, and evaluation document.

Besides following the MethOntology methodology [Fernández-López et al., 1997] during the LIoPY's building process, our major design consideration is to propose a lightweight and complete privacy ontology for the IoT domain. To this end, we design our ontology based on the following four principles:

- **Lightweight**: lightweight ontology model helps its adoption and reuse in other projects.
- **Completeness**: the ontology needs to integrate and extend the well-designed existing ontologies to ensure coverage of a large portion of the IoT domain.
- **Compatibility**: the ontology needs to be consistent with the existing ontologies in order to ensure compatibility.
- **Modularity**: the ontology is developed with a modular approach in order to ease its evolution, extension, and integration with other existing ontologies.

In the next sections, we only describe the new added concepts and properties of our proposed ontology without detailing the concepts that are imported from existing models.

2.3.2 LIoPY's Modules

To be compliant with today ontology engineering, we adopt the modularization method, which consists in segmenting an ontology into smaller parts [Haller et al., 2017]. The main feature of an ontology module is that it is self-contained, which means that the result of a query answering within a single module should be the same answers without the need to access other modules of the ontology [Haller et al., 2017].

Figure 2.9 depicts the different LIoPY's modules. In order to cover the whole IoT privacy aspects, LIoPY contains three main modules, namely IoT description, IoT resource management, and IoT resource result sharing management.

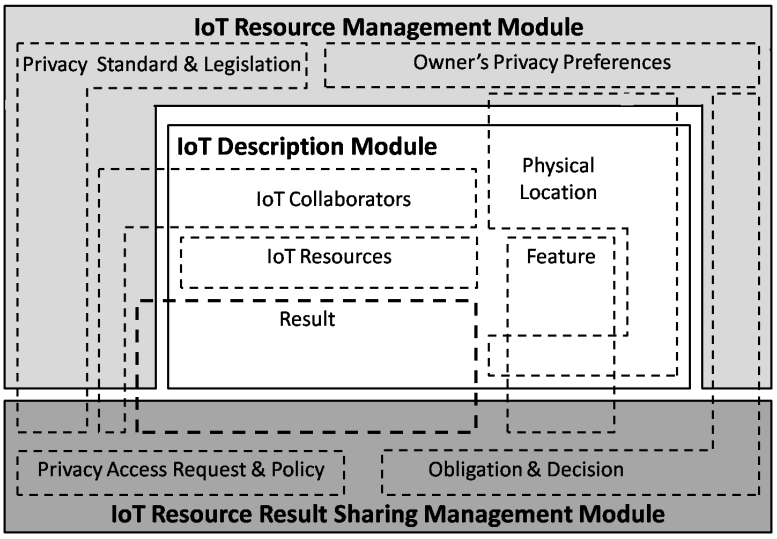


Figure 2.9: LIoPY's Modules

Each module includes a set of sub-modules and it aims at providing users of LIoPY with the knowledge they require, reducing the scope as much as possible to what is strictly necessary in a given use case as follows:

2.3.2.1 IoT Description Module

The IoT Description module deals with the IoT environment and includes five sub-modules, namely IoT Collaborators, IoT Resources, Result, Physical Location, and Feature. It describes the different collaborators into an IoT network and the feature of the IoT resources as well as the data generated by these resources. To do so, we reuse some classes and properties of the SSN [Haller et al., 2018] and IoT-lite [Bermudez-Edo et al., 2017] ontologies and extend them with some new terms in order to define features of the IoT resources and their outputs. Figure 2.10 shows the defined classes of the IoT description module and the relationships among them, namely Actor, IoT_Device, and IoT_Device_Output classes. For instance, the Data_Category class is proposed in order to help the data owner to classify the own IoT resource outputs. According to the data category of the IoT resource output, the data owner's privacy preferences will be enforced during the IoT data sharing phase.

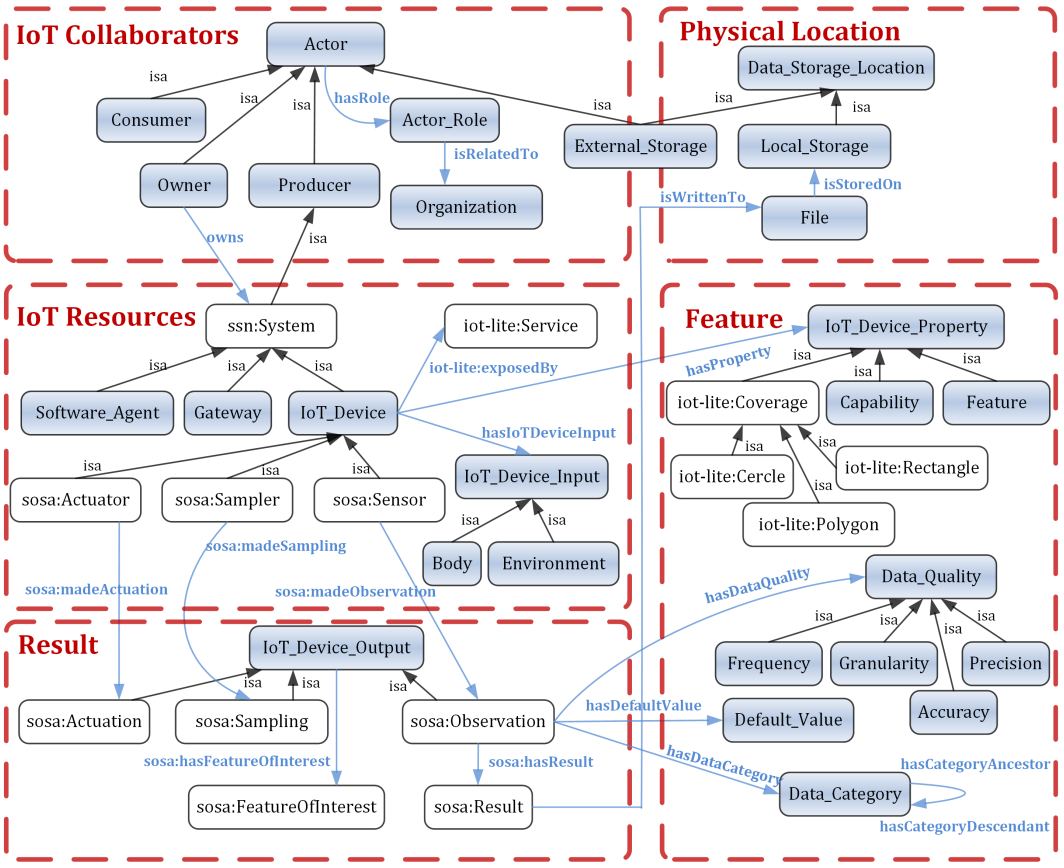


Figure 2.10: IoT Description Module

2.3.2.2 IoT Resource Management Module

The IoT Resource Management module helps the data owner to better control the IoT resources and their results. To this aim, it includes two sub-modules, namely Privacy Standard and Legislation as well as Owner's Privacy Preferences. Figure 2.11 *a)* shows the defined classes of the IoT resource management module, namely Privacy_Attribute, Privacy_Rule, and Privacy_Permission_Setting classes and the relationships among them. The Privacy_Attribute class has a set of sub-classes, namely Consent, Purpose, Retention, Operation, Condition, and Disclosure. These sub-classes specify respectively the data owner's awareness, for what reason, for how long, how, under which conditions the owner's data will be handled, and to whom they can be disclosed. These privacy attributes are defined in each Privacy_Rule class using the object properties hasAllowedIntendedPrivacyAttribute and hasProhibitedIntendedPrivacyAttribute. Moreover, the proposed Privacy_Permission_Setting class expresses the data owner privacy preferences on how the smart devices must behave. To this end, one or more permission settings can be associated with each device output using the object property hasPrivacyPermissionSetting.

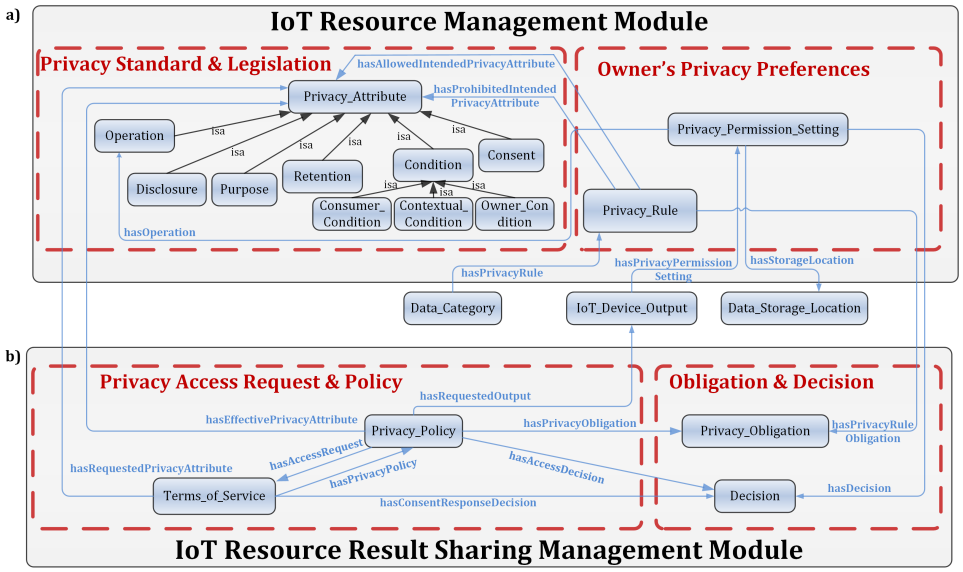


Figure 2.11: a) IoT Resource Management Module, b) IoT Resource Result Sharing Management Module

After presenting the second LIoPY module, we look at the last LIoPY module, which aims at preserving privacy from a shared-data-centric perspective.

2.3.2.3 IoT Resource Result Sharing Management Module

The IoT Resource Result Sharing Management module presents how the data must be handled once shared. To this aim, it includes two sub-modules, namely Privacy Access Request and Policy as well as Obligation and Decision. Figure 2.11 *b)* shows the defined classes of the IoT resource result sharing management module, namely Privacy_Policy, Terms_of_Service, and Privacy_Obligation classes and the relationships among them. The proposed Terms_of_Service class shares the same privacy attribute set with the privacy rule to generate a Privacy_Policy that matches the consumer's terms of service and the owner's privacy preferences. Each privacy policy has the hasRequestedOutput object property that defines the requested data by the consumer, the hasAccessDecision object property that defines if the requested output can be shared with the consumer, and the hasPrivacyObligation object property that defines a set of privacy obligations, such as data aggregation, data anonymization, and noise addition. These privacy obligations are chosen according to the data category of the data owner's IoT device outputs.

We propose below a privacy preferences model, which is based on the LIoPY's concepts and properties.

2.4 Privacy Preferences Model

In this section, we address the privacy issue in the IoT domain through two privacy-preserving perspectives, namely (i) collected-data-centric perspective that focuses on issues related to the collected data by the IoT devices and how these latter must behave and (ii) shared-data-centric

perspective that focuses on issues related to the shared data and how they must be handled by the consumers.

In order to facilitate understanding of LIoPY's concepts and relationships, we consider a simple scenario for the rest of this chapter. Let Alice, a 40-year old woman who owned a set of smart devices, namely a wireless body sensor, a smart treadmill, and a computer tablet. These smart devices continuously check her health conditions by measuring her heart rate, position, and steps during sport training. Some of Alice's produced data can be sent to the hospital and remotely monitored by Alice's doctor. In this context, we introduce below some examples for managing smart devices' settings and controlling the shared IoT data using LIoPY's concepts. Although, we illustrate our ideas in the healthcare context, our ontology is agnostic and can be applied in other IoT contexts.

2.4.1 Collected-data-centric privacy-preserving perspective

According to [Clarke, 2006], the privacy of the person is the right to control the integrity of the body. It covers physical requirements, health problems, and required wearable IoT devices. In order to guarantee this right, our ontology enables the definition of some settings to manage smart devices and their produced results.

IoT device output is the result generated by smart devices in the IoT domain. In LIoPY, the IoT_Device_Output class has a set of privacy permission settings. The Privacy_Permission_Setting class expresses the data owner privacy preferences on how the IoT devices must behave when collecting their results. To this end, one or more permission settings can be associated with each device output using the object property hasPrivacyPermissionSetting. For instance, in the healthcare domain, the device outputs can be vital signs that are collected by wireless medical sensors, such as blood sugar, pressure or heart rate.

Thus, we represent the Privacy_Permission_Setting class as a tuple of the following form:

$$Privacy\_Permission\_Setting =< hasOperation, operation\_frequency, hasStorageLocation, operation\_beginning\_time, operation\_ending\_time, hasDecision >$$

where:

- hasOperation: this is an object property that its value is one of a predefined set of individuals of the Operation class that can be Read_Operation, Write_Operation, Monitor_Operation, etc.
- operation_frequency: this is a data property that aims at fixing the occurrence rate of the execution of a given operation. For example, a data owner can allow writing on local storage every ten minutes.
- hasStorageLocation: this is an object property that its value is an individual of the Data_Storage_Location class, which includes External_Storage and Local_Storage sub-classes. For example, a data owner can disallow writing the own location on the cloud storage.

- operation_beginning_time, operation_ending_time: these are data properties that aim at fixing the time to start and finish the operation execution. For example, a data owner can allow collecting the own location only for a specific period of time.
- hasDecision: this is an object property that its value can be Permit or Deny.

These privacy permission settings are defined before the beginning of the collection phase. When buying a new smart device, the data owner can define the privacy permission setting preferences about each device output via a portal in the own gateway.

For instance, Alice practices some sport activities at home using her smart treadmill. This latter can collect Alice's steps and her training duration. Figure 2.12 depicts an example of a privacy permission setting instance, called Treadmill_Privacy_Permission_Setting associated with Alice_step. The defined instance authorizes to collect Alice's steps during her training and add the new produced data to Alice_Local_Storage in order to be locally saved.

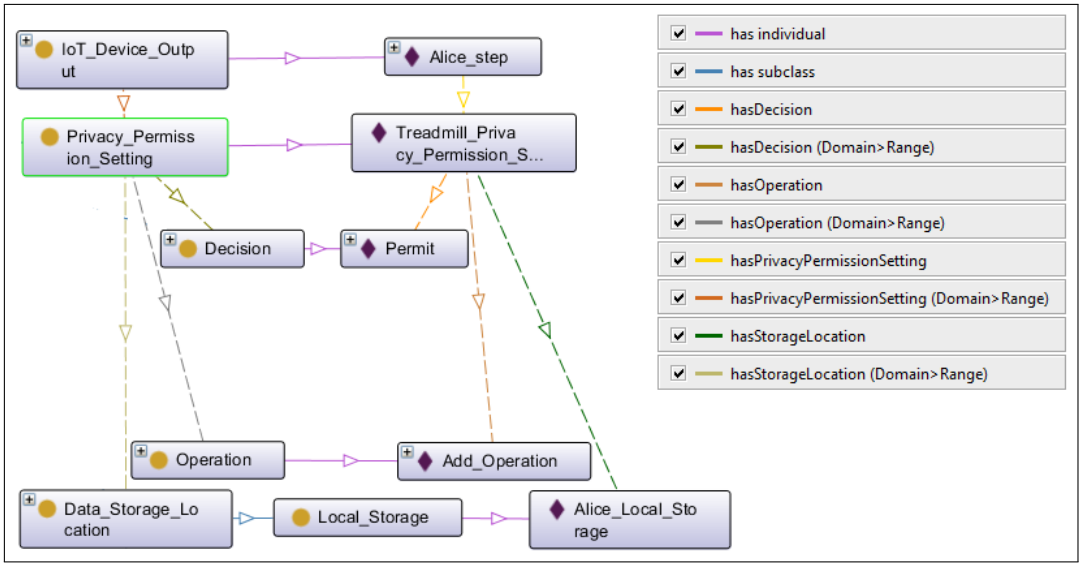


Figure 2.12: Example of a privacy permission setting instance

These permission settings will be locally stored and regularity verified before allowing any device to communicate with other devices or connect to the Internet. The permission setting verification leads to enforce the data owner control on the own devices and rapidly detect any malicious attempt by analyzing the device behavior.

After presenting LIoPY's terms that guarantee the owner's right to control and manage the owned smart devices, we introduce below LIoPY's terms that guarantee the owner's right to control and manage the IoT data handling.

2.4.2 Shared-data-centric privacy-preserving perspective

According to [Clarke, 2006], the privacy of personal information involves the right to control when, where, how, to whom, and to what extent an individual shares the own personal information. In order to guarantee these rights, our ontology enables the definition of some privacy

requirements between both the data owner and the data consumer in order to match the privacy preferences and promises to create a common privacy policy.

Figure 2.13 depicts the five LIoPY's classes that are involved in the privacy policy generation. These classes are Privacy_Attribute, Data_Category, Privacy_Rule, Terms_of_Service, and Privacy_Policy classes.

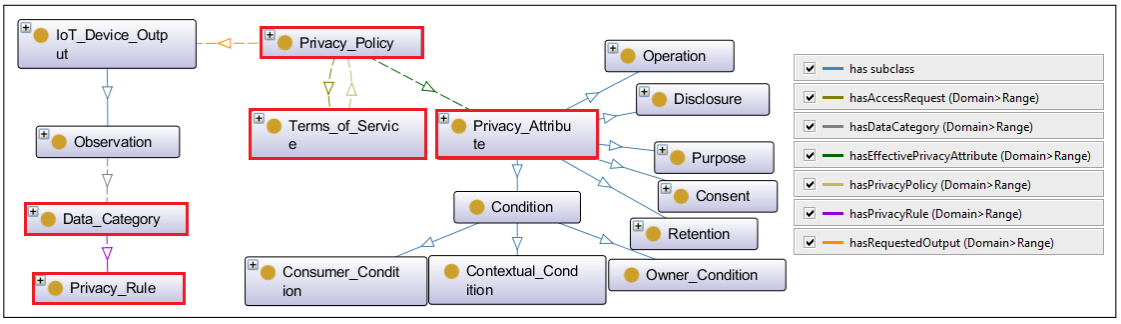


Figure 2.13: LIoPY's principal classes

A description of the ontological concepts is provided on the following subsections.

2.4.2.1 Privacy_Attribute class

Privacy involves the right to control when, where, how, to whom, and to what extent an individual shares the own personal information [Clarke, 2006]. Besides, the Organization for Economic Co-operation and Development (OECD) [OECD, 1981] and the ISO standard [ISO/IEC29100, 2011] provided a set of principles that enable individuals to express their privacy requirements and place obligations on organizations to follow. Moreover, the European Regulation [GDPR, 2016] embedded these principles.

Based on the privacy definition and the studied principles, we define the Privacy_Attribute class. In LIoPY, we consider Consent, Purpose, Retention, Operation, Condition, and Disclosure as sub-classes of the Privacy_Attribute class. These sub-classes specify respectively the data owner's awareness, for what reason, for how long, how, under which conditions the owner's data will be handled, and to whom they can be disclosed.

More specifically, the Privacy_Attribute class presents different constraints according to the data owner's privacy preferences and the data consumer's terms of service. This class is defined using the following form:

$$CONSENT \sqsubseteq \text{Privacy\_Attribute}; PURPOSE \sqsubseteq \text{Privacy\_Attribute};$$
$$RETENTION \sqsubseteq \text{Privacy\_Attribute}; OPERATION \sqsubseteq \text{Privacy\_Attribute};$$
$$CONDITION \sqsubseteq \text{Privacy\_Attribute}; DISCLOSURE \sqsubseteq \text{Privacy\_Attribute};$$

The Privacy_Attribute is a set of disjoint sub-classes. For instance, the Treatment_Purpose is only an instance of the Purpose class and cannot be an instance of the other Privacy_Attribute sub-classes. Disjointedness axiom about two classes states that they cannot share a same individual:

$$CONSENT \sqcap PURPOSE \sqcap RETENTION \sqcap OPERATION$$
$$\sqcap CONDITION \sqcap DISCLOSURE \sqsubseteq \bot;$$

Besides, the Privacy_Attribute is the union of all the sub-classes. Therefore, each individual should not be a direct individual of the Privacy_Attribute class but an individual of only one subclass, hence no shared individuals are allowed:

$$CONSENT \sqcup PURPOSE \sqcup RETENTION \sqcup OPERATION \sqcup CONDITION \sqcup DISCLOSURE \equiv \text{Privacy\_Attribute};$$

We detail each of the Privacy_Attribute sub-classes on the following subsections.

Consent

According to the GDPR [GDPR, 2016], personal data cannot be collected or used by the consumers without acceptance of the data owner.

Definition 1: Consent. Consent specifies the data owner's awareness concerning the data collection and data access by the data consumers.

Figure 2.14 depicts an example of the Consent class instantiation. Thus, the predefined set of individuals of the Consent class has three possible individuals, namely NeedForConsent, which means the obligation of an explicit acceptance of the data owner before collecting and using the IoT data, NeedForParentalConsent, which means the obligation of an explicit acceptance of the data owner's parent in case of a minor data owner, and NoNeedForConsent, which means no need for an explicit data owner's acceptance.

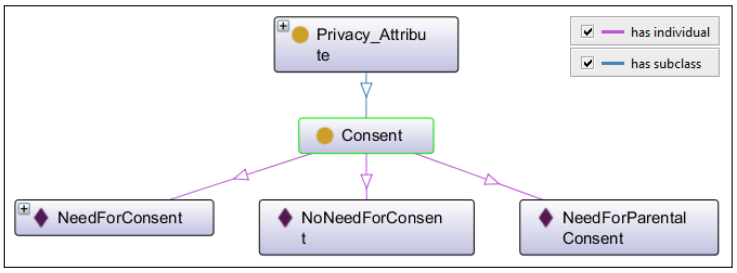


Figure 2.14: Example of the Consent class instantiation

Definition 2: Intended Consent. Intended consent specifies the data owner's preferences acceptance or rejection to use the IoT data. An intended consent is denoted as *ic*.

Definition 3: Consent Response Decision. Consent response decision specifies the data owner's decision to share or not the collected data with a specific data consumer. A consent response decision denoted as *crd* with $crd \in \{\text{Permit}, \text{Deny}\}$.

Definition 4: Requested Consent Compliance. Let *ic* and *crd* the intended consent and the consent response decision, respectively. The data consumer's request is said to be consent compliant to the intended consent *ic*, only if one of the following condition is satisfied:

1. *ic* is *sameAs* "NoNeedForConsent"

2. *ic* is *sameAs* "NeedForConsent" AND *crd* is *sameAs* "Permit"

3. *ic* is *sameAs* "NeedForParentalConsent" AND *crd* is *sameAs* "Permit"

Purpose

Purpose class specifies the reasons for which the owner's data can be accessed. In fact, according to the purpose, the granularity of the data and the privacy-preserving mechanisms that will be applied to the data before the sharing task can be specified. In order to simplify the management, purposes are organized according to a hierarchical structure based on the principles of generalization and specialization.

Definition 1: Purpose and Purpose Tree. Purpose specifies the reasons for data collection and data access. The collection of purposes defined for a specified scenario forms a purpose set, denoted as $\mathcal{P}$. This set of purposes is organized hierarchically into a tree, denoted as $\mathcal{PT}$. Each node of the purpose tree is a purpose, denoted as p_i with $p_i \in \mathcal{P}$ and $\mathcal{P} \in \mathcal{PT}$. A purpose p_i is modeled as a pair <$\mathcal{A}ncp_i$, $\mathcal{D}esp_i$>, where $\mathcal{A}ncp_i$ are the ancestors of p_i into the purpose tree including p_i itself, whereas $\mathcal{D}esp_i$ are the descendants of p_i into the purpose tree including p_i itself.

Figure 2.15 depicts the purposes as a tree structure where each predefined individual of the Purpose class is related to other purposes by the hasPurposeDescendant object property that is a hierarchical relation with its descendants. In order to facilitate navigation in our ontology, we define the hasPurposeAncestor object property as an *inverseOf* the hasPurposeDescendant one. For simplicity purposes, the hasPurposeAncestor flows have been hidden in figure 2.15.

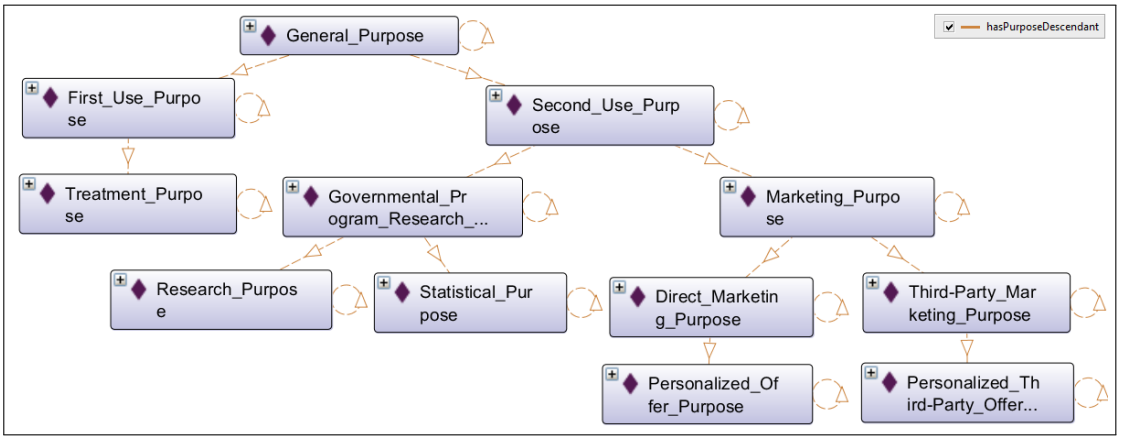


Figure 2.15: Example of a purpose tree

Definition 2: Intended Purpose. Intended purpose specifies the reasons for which the shared data can or cannot be collected and accessed by consumers according to the data owner's preferences. An intended purpose, denoted as ip is modeled as a pair <$\mathcal{A}ip$, $\mathcal{P}ip$>, where $\mathcal{A}ip \subseteq \mathcal{P}$ is a set of allowed intended purposes and $\mathcal{P}ip \subseteq \mathcal{P}$ is a set of prohibited intended purposes. The conflicts between the allowed intended purposes and the prohibited intended

purposes for the same data are resolved by assuming that prohibited intended purposes override allowed intended ones.

Definition 3: Requested Purpose. Requested Purpose specifies the reasons for which the shared data are collected and accessed according to the data consumer's terms of service. A requested purpose is denoted as rp with $rp \in \mathcal{P}$ and $\mathcal{P} \in \mathcal{PT}$.

Definition 4: Requested Purpose Compliance. Let ip=<$\mathcal{A}ip$, $\mathcal{P}ip$> and rp, where $ip \in \mathcal{P}$, $rp \in \mathcal{P}$ and $\mathcal{P} \in \mathcal{PT}$. The requested purpose rp is said to be compliant to the intended purpose ip into the purpose tree $\mathcal{PT}$, only if:

1. $rp \notin (\cup_{pip_k \in Pip}\ \mathcal{A}nc_{ip_k}) \cup (\cup_{pip_k \in Pip}\ \mathcal{D}es_{ip_k})$, which means that the requested purpose rp dœs not belong to the union of the prohibited intended purpose ancestors and the prohibited intended purpose descendants.
2. $rp \in \cup_{aip_j \in Aip}\ \mathcal{D}es_{ip_j}$, which means that the requested purpose rp belongs to the allowed intended purpose descendants.

Retention

The GDPR [GDPR, 2016] states that personal data collected and stored within a European Union country should be stored for a reasonable time duration.

Definition 1: Retention. Retention specifies the duration of the data collection, storage, and/or data processing by the data consumers.

Figure 2.16 depicts an example of a retention individual. Retention class has a data property, called retention_duration_per_day, which defines the time duration for storing data by the data consumers. Thus, the predefined set of individuals of the Retention class aims at limiting the lifetime of the shared data to fulfill the specified purpose as long as necessary and thereafter securely destroying the data.

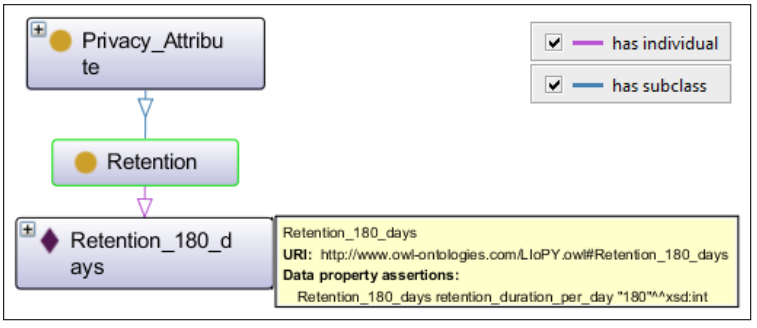


Figure 2.16: Example of a retention individual

Definition 2: Intended Retention. Intended retention specifies the time duration of the data collection, storage, and/or data processing by consumers according to the data owner's preferences. An intended retention is denoted as ir.

Definition 3: Requested Retention. Requested retention specifies the desired time duration of the data collection, storage, and/or data processing according to the data consumer's terms of service. A requested retention is denoted as *rr*.

Definition 4: Requested Retention Compliance. Let *ir*, *rr*, and retention_duration_per_day the requested retention, the intended retention, and the time duration for storing data, respectively. The requested retention *rr* is said to be compliant to the intended retention *ir*, only if:

- rr.retention_duration_per_day $\leq$ ir.retention_duration_per_day

Operation

Operation class specifies the usage or actions that can be applied to the shared data. In order to simplify the management, operations are organized according to a hierarchical structure based on the principles of generalization and specialization.

Definition 1: Operation and Operation Tree. Operation specifies the allowed usage or actions on the shared data. The collection of operations defined for a specified scenario forms an operation set, denoted as $\mathcal{O}$. This set of operations is organized hierarchically into a tree, denoted as $\mathcal{OT}$. Each node of the operation tree is an operation, denoted as o_i with $o_i \in \mathcal{O}$ and $\mathcal{O} \in \mathcal{OT}$. An operation o_i is modeled as a pair <$\mathcal{A}nco_i$, $\mathcal{D}eso_i$>, where $\mathcal{A}nco_i$ are the ancestors of o_i into the operation tree including o_i itself, whereas $\mathcal{D}eso_i$ are the descendants of o_i into the operation tree including o_i itself.

Figure 2.17 depicts the operations as a tree structure where each predefined individual of the Operation class is related to other operations by both the hasOperationAncestor hasOperationDescendant object properties.

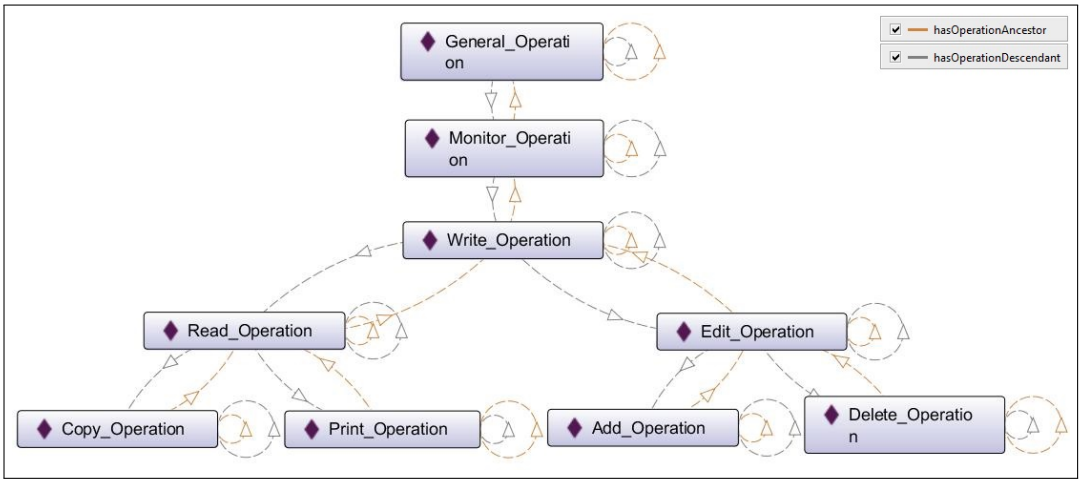


Figure 2.17: Example of an operation tree

Definition 2: Intended Operation. Intended operation specifies the actions that can or cannot be applied to the shared data by consumers according to the data owner's preferences. An

intended operation, denoted as *io* is modeled as a pair <$\mathcal{A}io$, $\mathcal{P}io$>, where $\mathcal{A}io \subseteq \mathcal{O}$ is a set of allowed intended operations and $\mathcal{P}io \subseteq \mathcal{O}$ is a set of prohibited intended operations.

Definition 3: Requested Operation. Requested operation specifies the desired usage according to the data consumer's terms of service. A requested operation is denoted as *ro* with $ro \in \mathcal{O}$ and $\mathcal{O} \in \mathcal{OT}$.

Definition 4: Requested Operation Compliance. Let *io*=<$\mathcal{A}io$, $\mathcal{P}io$> and *ro*, where $io \in \mathcal{O}$, $ro \in \mathcal{O}$ and $\mathcal{O} \in \mathcal{OT}$. The requested operation *ro* is said to be compliant to the intended operation *io* into the operation tree $\mathcal{OT}$, only if:

1. $ro \notin (\cup_{pio_k \in Pio}\ \mathcal{A}nc_{io_k}) \cup (\cup_{pio_k \in Pio}\ \mathcal{D}es_{io_k})$, which means that the requested operation *ro* dœs not belong to the union of the prohibited intended operation ancestors and the prohibited intended operation descendants.

2. $ro \in \cup_{aio_j \in Aio}\ \mathcal{D}es_{io_j}$, which means that the requested operation *ro* belongs to the allowed intended operation descendants.

Condition

Condition class specifies a set of requirements that needs to be satisfied by each involved party for the data sharing.

Definition 1: Condition. Condition specifies a set of requirements for sharing the collected data. The condition sub-classes forms a condition set, denoted as $\mathcal{C}ond = \{\text{Consumer\_Condition} \cup \text{Contextual\_Condition} \cup \text{Owner\_Condition}\}$. Each node of the condition set is a condition, denoted as $cond_i$ with $cond_i \in \mathcal{C}ond$.

Figure 2.18 depicts the conditions as a tree structure. In our case, the condition can refer to the data Owner_Condition (e.g., country and age of the data owner), the Consumer_Condition (e.g., consumer's role or technical capabilities, such as cryptographic protocols), or the Contextual_Condition (e.g., normal or emergency situations).

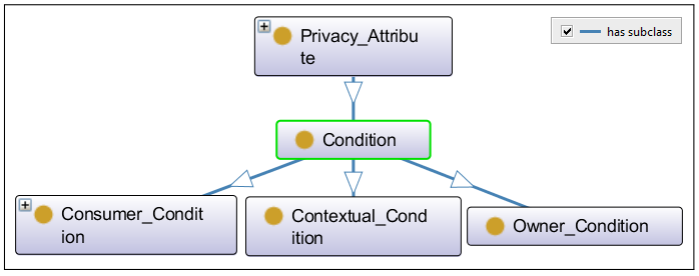


Figure 2.18: Condition class: Ancestor and Descendants

Definition 2: Intended Condition. Intended condition specifies the required situation that needs to be satisfied by the data consumer according to the data owner's preferences. An intended condition, denoted as *icond* is modeled as a pair <$\mathcal{A}icond$, $\mathcal{P}icond$>, where

- $\mathcal{A}icond \subseteq$ (Consumer_Condition $\cup$ Contextual_Condition) is a set of allowed intended conditions. For instance, the 'emergency medical records' are accessible when the consumer's role is 'emergency service crew' and the contextual condition is 'emergency situation'.

- $\mathcal{P}icond \subseteq$ (Consumer_Condition $\cup$ Contextual_Condition) is a set of prohibited intended conditions. For instance, the 'emergency medical records' are not accessible when the consumer's role is 'emergency service crew' and the contextual condition is 'normal situation'.

Definition 3: Requested Condition. Requested condition specifies the desired condition that the data owner needs to satisfy according to the data consumer's terms of service. Requested conditions are denoted as *rcond* with *rcond* $\subseteq$ Owner_Condition.

Definition 4: Intended Condition Compliance. Let *ar* an allowed role defined by the intended condition, denoted as *icond*, *init* an initiator of terms of service, and *role* an initiator's role. The terms of service *tos* are said to be compliant to the intended condition *icond*, only if:

- $tos.init.role \equiv icond.ar$, which means that the terms of service initiator's role is similar to the allowed role defined by the intended condition.

Disclosure

Disclosure class specifies the data transfer requirements that can be applied to the shared data. In order to simplify the management, disclosure types are organized according to a hierarchical structure based on the principles of generalization and specialization.

Definition 1: Disclosure and Disclosure Tree. Disclosure specifies the allowed parties that can receive the collected data. The collection of disclosure types defined for a specified scenario forms a disclosure set, denoted as $\mathcal{D}isc$. This set of disclosure types is organized hierarchically into a tree, denoted as $\mathcal{DT}$. Each node of the disclosure tree is a disclosure type, denoted as $disc_i$ with $disc_i \in \mathcal{D}isc$ and $\mathcal{D}isc \in \mathcal{DT}$. A disclosure type $disc_i$ is modeled as a pair <$\mathcal{A}ncdisc_i$, $\mathcal{D}esdisc_i$>, where $\mathcal{A}ncdisc_i$ are the ancestors of $disc_i$ into the disclosure tree including $disc_i$ itself, whereas $\mathcal{D}esdisc_i$ are the descendants of $disc_i$ into the disclosure tree including $disc_i$ itself.

According to the GDPR [GDPR, 2016], without explicit acceptance of the data owner, personal data should not be disclosed to third-parties. Thus, the predefined set of individuals of the Disclosure class can be used to ensure law enforcement. Figure 2.19 depicts the disclosure types as a tree structure where each predefined individual of the Disclosure class is related to other disclosure types by both the hasDisclosureAncestor and hasDisclosureDescendant object properties. For instance, the law stipulates that there are several data disclosure types. The default type is the one that dœs not allow data transmission to any party, namely No_Sharing. With_Everyone is the second type that consists in data transfer to any third-party asking for the data, such as data sharing With_Consumer_Partners, With_Legal_Parties, or With_Public_Third_Parties. The disclosure type, called With_Consumer_Only enables only the data consumer to use the data.

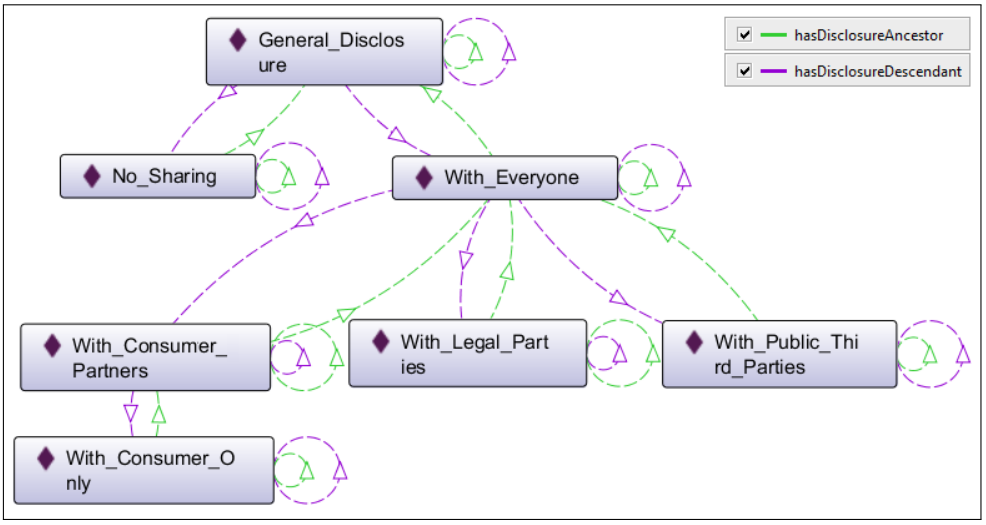


Figure 2.19: Example of a disclosure tree

Definition 2: Intended Disclosure. Intended disclosure specifies the allowed parties that can receive the collected data according to the data owner's preferences. An intended disclosure, denoted as *idisc* is modeled as a pair <$\mathcal{A}idisc$, $\mathcal{P}idisc$>, where $\mathcal{A}idisc \subseteq \mathcal{D}isc$ is a set of allowed intended disclosures and $\mathcal{P}idisc \subseteq \mathcal{D}isc$ is a set of prohibited intended disclosures.

Definition 3: Requested Disclosure. Requested disclosure specifies the desired parties that the data consumer wants to share with them the collected data according to the data consumer's terms of service. A requested disclosure is denoted as *rdisc* with $rdisc \in \mathcal{D}isc$ and $\mathcal{D}isc \in \mathcal{DT}$.

Definition 4: Requested Disclosure Compliance. Let *idisc*=<$\mathcal{A}idisc$, $\mathcal{P}idisc$> and *rdisc*, where $idisc \in \mathcal{D}isc$, $rdisc \in \mathcal{D}isc$ and $\mathcal{D}isc \in \mathcal{DT}$. The requested disclosure *rdisc* is said to be compliant to the intended disclosure *idisc* into the disclosure tree $\mathcal{DT}$, only if:

1. $rdisc \notin (\cup_{pidisc_k \in Pidisc} \mathcal{A}nc_{idisc_k}) \cup (\cup_{pidisc_k \in Pidisc} \mathcal{D}es_{idisc_k})$, which means that the requested disclosure *rdisc* dœs not belong to the union of the prohibited intended disclosure ancestors and the prohibited intended disclosure descendants.

2. $rdisc \in \cup_{aidisc_j \in Aidisc} \mathcal{D}es_{idisc_j}$, which means that the requested disclosure *rdisc* belongs to the allowed intended disclosure descendants.

After detailing the Privacy_Attribute class and its sub-classes, we present in the following section the Data_Category class.

2.4.2.2 Data_Category class

Data_Category class specifies the type of the IoT device outputs. The IoT device outputs can be classified into several data categories according to the data owner's privacy preferences. In order to simplify the management, data categories are organized according to a hierarchical structure based on the principles of generalization and specialization.

Definition 1: Data category and Data category Tree. Data category specifies the type of information of the collected IoT data. Categories are organized hierarchically into a tree, denoted as $\mathcal{CT}$ that is a category set, denoted as $\mathcal{C}$. Each node of the category tree is a data category, denoted as dc_i with $dc_i \in \mathcal{C}$ and $\mathcal{C} \in \mathcal{CT}$.
A data category dc_i is modeled as a pair <$\mathcal{A}ncdc_i$, $\mathcal{D}esdc_i$>, where $\mathcal{A}nsdc_i$ are the ancestors of dc_i into the data category tree including dc_i itself, whereas $\mathcal{D}esdc_i$ are the descendants of dc_i into the data category tree including dc_i itself.

Considering the privacy guidelines [OECD, 1981], data protection laws [Directive 95/46/EC, 1995] [GDPR, 2016], privacy framework [ISO/IEC29100, 2011], and literature on privacy publishing [Benjamin et al., 2010], we propose the following data categories:

- **Explicit-Identifier**: this is a set of attributes, such as social security number (SSN) and name, containing information that explicitly identifies tuple owners.
- **Quasi-Identifier**: this is a set of attributes that can potentially identify tuple owners, such as birthday, gender, etc.
- **Sensitive-Attributes**: this consists of sensitive information such as salary, physical health and psychological health status.
- **Non-Sensitive-Attributes**: this contains all attributes that do not fall into the previous categories.

Figure 2.20 depicts an example of a data category tree where each predefined individual of the Data_Category class is related to other data categories by the hasCategoryAncestor object property that is a hierarchical relation with its ancestors. Each category is an ancestor of itself and has a second object property, called hasCategoryDescendant that is the *inverseOf* hasCategoryAncestor.

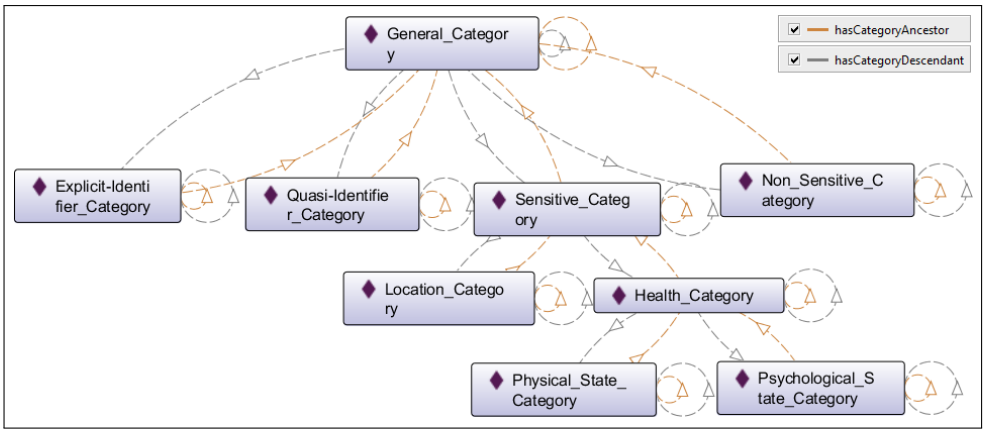


Figure 2.20: Example of a data category tree

Definition 2: Intended data category. Intended data category specifies a set of data categories that can or cannot be derived from the requested data. An intended data category, denoted as *idc* is modeled as a pair <$\mathcal{A}idc$, $\mathcal{P}idc$>, where $\mathcal{A}idc \subseteq \mathcal{C}$ is a set of allowed intended data categories and $\mathcal{P}idc \subseteq \mathcal{C}$ is a set of prohibited intended data categories.

Definition 3: Requested data category. Requested data category specifies the type of the requested data according to the data consumer's terms of service. A requested data category is denoted as *rdc* with $rdc \in \mathcal{C}$ and $\mathcal{C} \in \mathcal{CT}$.

Definition 4: Requested data category Compliance. Let *idc*=<$\mathcal{A}idc$, $\mathcal{P}idc$> and *rdc*, where $idc \in \mathcal{C}$, $rdc \in \mathcal{C}$ and $\mathcal{C} \in \mathcal{CT}$. The requested data category *rdc* is said to be compliant to the intended data category *idc* into the data category tree $\mathcal{CT}$, only if:

1. $rdc \notin (\cup_{pidc_k \in Pidc}\ \mathcal{A}nc_{idc_k}) \cup (\cup_{pidc_k \in Pidc}\ \mathcal{D}es_{idc_k})$, which means that the requested data category *rdc* dœs not belong to the union of the prohibited intended data category ancestors and the prohibited intended data category descendants.
2. $rdc \in \cup_{aidc_j \in Aidc}\ \mathcal{D}es_{idc_j}$, which means that the requested data category *rdc* belongs to the allowed intended data category descendants.

In order to help the data owner to control the own IoT data, we propose to preserve each data category with a privacy rule, which expresses the data owner's privacy preferences.

2.4.2.3 Privacy_Rule class

Privacy_Rule class helps the data owner to define some privacy preferences in order to preserve the collected data privacy. It specifies how the shared data must be handled by the data consumers. To this end, one or more rules can be associated with each device output data category using the object property hasPrivacyRule. Because the data owner generally has not enough expertise in the privacy domain to choose the appropriate privacy rule for each IoT device output, we choose to define rules for each data category. Then, the IoT device output inherits the privacy rules according to its data category. Thus, the data owner need only to define the category of the own data.

Definition 1: Intended privacy attribute. Intended privacy attribute specifies a set of privacy attributes that is defined on one privacy rule. An intended privacy attribute, denoted as *intendedPA* is modeled as a tuple <*ic*, *ip*, *ir*, *io*, *idisc*, *icond*>, where *ic* is an intended consent, *ip* an intended data use purpose, *ir* an intended retention time, *io* an intended operation, *idisc* an intended data disclosure, and *icond* an intended consumer's condition and contextual condition to be satisfied to apply the data owner privacy rule. Each privacy attribute value belongs to the set of its predefined values.

Definition 2: Privacy Rule. Privacy rule specifies the data owner's privacy preferences about how the shared data must be handled by the data consumers. We represent the Privacy_Rule class, as a tuple of the following form:

$$Privacy\_Rule =< hasAllowedIntendedPrivacyAttribute, hasProhibitedIntendedPrivacyAttribute, hasPrivacyRuleObligation >$$

The hasAllowedIntendedPrivacyAttribute and hasProhibitedIntendedPrivacyAttribute object properties enable each privacy rule to include the privacy attributes that are defined by the data owner and can regulate the data handling by consumers to preserve the data owner's privacy. The domain of both hasAllowedIntendedPrivacyAttribute and hasProhibitedIntendedPrivacyAttribute object properties is the Privacy_Rule class and their range is the Privacy_Attribute class, with:

- $Range_{\text{hasAllowedIntendedPrivacyAttribute}} \subseteq \cup_{ipa \in intendedPA} \mathcal{A}ipa$, which means that the hasAllowedIntendedPrivacyAttribute object property range values are included on the union of the allowed values of each intended privacy attributes.

- $Range_{\text{hasProhibitedIntendedPrivacyAttribute}} \subseteq \cup_{ipa \in intendedPA} \mathcal{P}ipa$, which means that the hasProhibitedIntendedPrivacyAttribute object property range values are included on the union of the prohibited values of each intended privacy attributes.

Furthermore, hasPrivacyRuleObligation object property associates for each privacy rule some privacy obligations. The Privacy_Rule class is the domain of hasPrivacyRuleObligation object property and its range is the Privacy_Obligation class. For instance, when the data owner wants to allow only the data consumer to use the data, a common key can be shared between both the data owner and the data consumer to ensure the shared data confidentiality. To this end, Symmetric_Encryption is one example of the security mechanisms that can be imposed by a Privacy_Rule.

Figure 2.21 depicts an example of a privacy rule instance defined for the sensitive data category, called Sensitive_Data_Privacy_Rule.

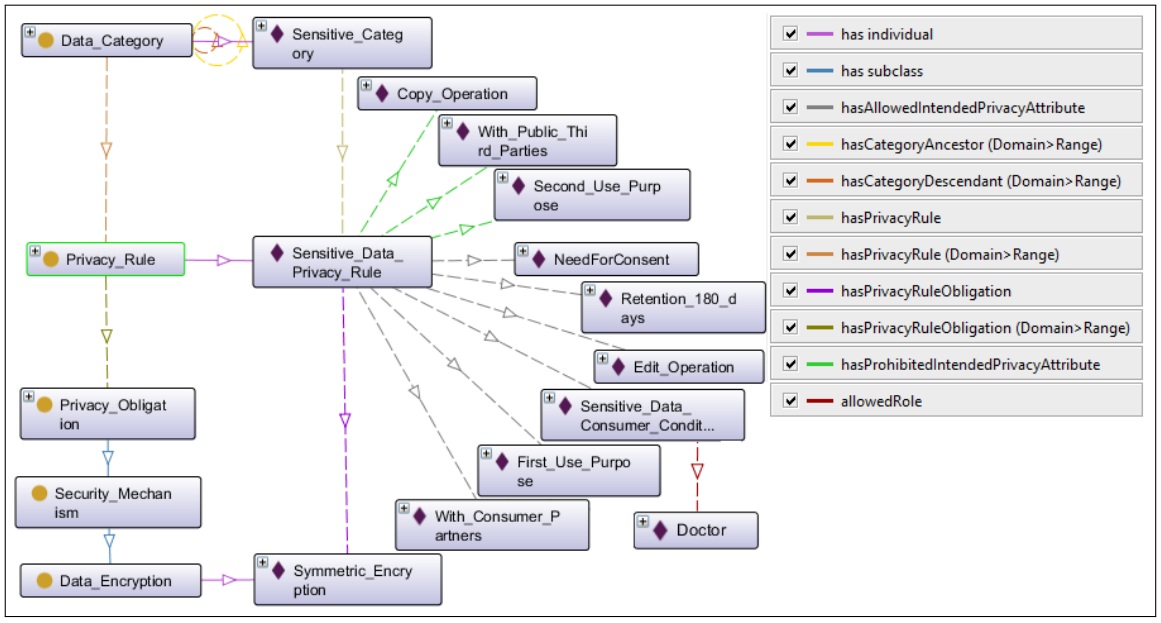


Figure 2.21: Example of a privacy rule instance

This privacy rule defines six allowed privacy attributes and three prohibited privacy attributes. IoT data that are protected by this rule need an explicit data owner's consent to be used for first use purpose, during six months, and can be edited only by a consumer whose

role is a doctor. The allowed consumer can disclosure the IoT data with some healthcare professional partners, such as nurse, member of the emergency staff, or physiotherapist. Moreover, IoT data that are protected by this rule cannot be copied, used for second use purpose, or disclosed to public third-parties. These three prohibitions are defined as the range of the object property hasProhibitedIntendedPrivacyAttribute, namely Copy_Operation, Second_Use_Purpose, and With_Public_Third_Parties.

While LIoPY enables the data owners to create fine-grained privacy preferences for their shared data, it also helps the data consumers to define some privacy promises thanks to the Terms_of_Service class.

2.4.2.4 Terms_of_Service class

Terms_of_Service class helps the data consumer to define its privacy promises in order to provide its service to the data owner. It specifies how the shared data will be handled by the data consumer's systems. To this end, one or more data types can be associated with the requested_data_name data property. These data types are the data requested by the data consumer to be collected, stored and processed.

Definition 1: Requested privacy attribute. Requested privacy attribute specifies a set of privacy attributes that is defined on terms of service. A requested privacy attribute, denoted as *requestedPA* is modeled as a tuple <$rp, rr, ro, rdisc, rcond$>, where rp a requested data use purpose, rr a requested retention time, ro a requested operation, $rdisc$ a requested data disclosure, and $rcond$ a requested owner's condition. Each privacy attribute value belongs to the set of its predefined values.

Definition 2: Terms of Service. Terms of service specify the data consumer's privacy promises about how the shared data will be handled by the data consumer systems in order to provide a service. We represent the Terms_of_Service class, as a tuple of the following form:

$$Terms\_of\_Service =< hasRequestedPrivacyAttribute, hasConsentResponseDecision, hasPrivacyPolicy >$$

The hasRequestedPrivacyAttribute object property enables each consumer's terms of service to include the defined privacy attributes that are mandatory to offer the consumer's service. The domain of hasRequestedPrivacyAttribute object property is the Terms_of_Service class and its range is the Privacy_Attribute class, with:

- $Range_{\text{hasRequestedPrivacyAttribute}} \subseteq requestedPA$, which means that the hasRequestedPrivacyAttribute object property range values are included on the set of the requested privacy attributes.

Moreover, the hasConsentResponseDecision object property specifies the data owner's decision to share or not the collected data with a specific data consumer. Its range is the Decision class.

The hasPrivacyPolicy object property defines how the data can be handled if the consumer's terms of service match the data owner's privacy preferences. Its domain is the Terms_of_Service class and its range is the Privacy_Policy class.

Figure 2.22 depicts an example of terms of service instance, called Terms_of_Service_for_Heartrate. The initiator of this terms of service instance is Bob, which is an individual of the Consumer class and has Doctor as a role. Four privacy attributes are defined as the range of the object property hasRequestedPrivacyAttribute, namely Treatment_Purpose, Edit_Operation, Retention_60_days, and With_Consumer_Only. Thus, the IoT data are requested for treatment purpose, to be edited during two months, and will not be disclosed.

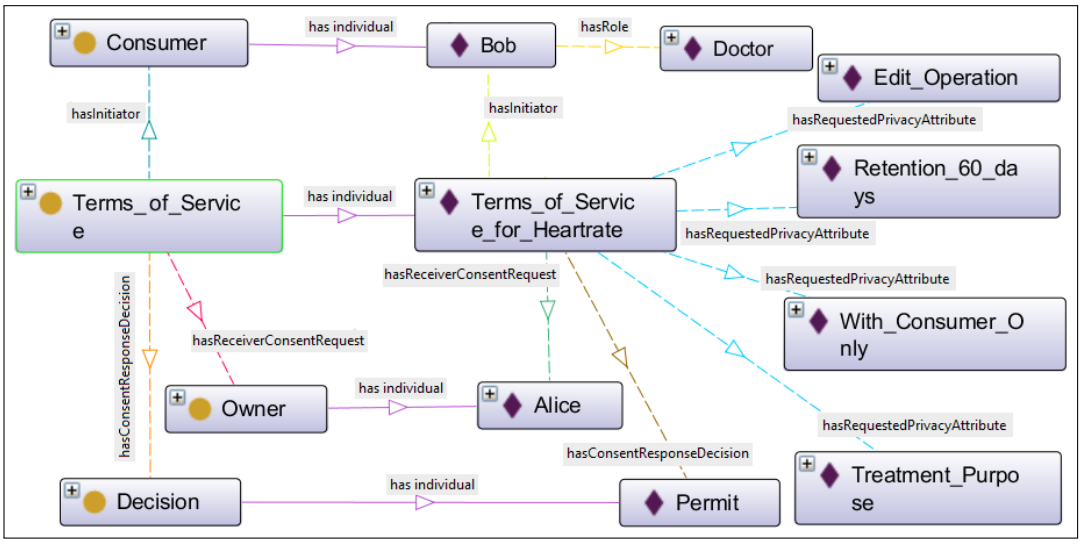


Figure 2.22: Example of terms of service instance

After presenting the main LIoPY's classes, namely Privacy_Attribute, Data_Category, Privacy_Rule, and Terms_of_Service that are involved in the privacy policy generation, we define below the Privacy_Policy class.

2.4.2.5 Privacy_Policy class

Privacy_Policy class is automatically inferred in case of a match between the data owner's privacy preferences and the consumer's terms of service.

Definition 1: Effective privacy attribute. Effective privacy attribute specifies a set of privacy attributes that are defined on one privacy policy. An effective privacy attribute, denoted as $effectivePA$ is modeled as a tuple $<ec, ep, er, eo, edisc, econd>$, where ec an effective consent, ep a requested data use purpose, er an effective retention time, eo an effective operation, $edisc$ an effective data disclosure, and $econd$ an effective owner's condition. Each privacy attribute value belongs to the set of its predefined values.

Definition 2: Privacy Policy. Privacy policy specifies the intersection between the data owner's privacy preferences and the data consumer's terms of service about how the data will be handled

once shared. We represent the Privacy_Policy class, as a tuple of the following form:

$$Privacy\_Policy =< hasEffectivePrivacyAttribute, hasPrivacyObligation, hasAccessDecision >$$

The hasEffectivePrivacyAttribute object property defines how the data will be handled once shared. The requested privacy attributes are added to the privacy policy in case of a match between the intended and the requested privacy attributes. Its domain is the Privacy_Policy class and its range is the Privacy_Attribute class, with:

- $Range_{\text{hasEffectivePrivacyAttribute}} \subseteq intendedPA \cap requestedPA$, which means that the hasEffectivePrivacyAttribute object property range values are included on the set of the intersection of the intended privacy attributes and the requested privacy attributes.

The hasPrivacyObligation object property associates for each privacy policy the privacy obligations according to the data category's privacy rule of the requested data. The Privacy_Policy class is the domain of hasPrivacyObligation object property and its range is the Privacy_Obligation class.

Moreover, the hasAccessDecision object property specifies the privacy policy's decision. Its range is the Decision class, which is instantiated as Permit or Deny.

Figure 2.23 shows an example of a privacy policy, called Privacy_Policy_Alice_Heartrate that is expected after matching the Sensitive_Data_Privacy_Rule (see Figure 2.21) and the Terms_of_Service_for_Heartrate (see Figure 2.22). Eight privacy attributes need to be generated as the range of the object property hasEffectivePrivacyAttribute. Moreover, the privacy policy needs to inherit the Symmetric_Encryption obligation from the aforementioned privacy rule.

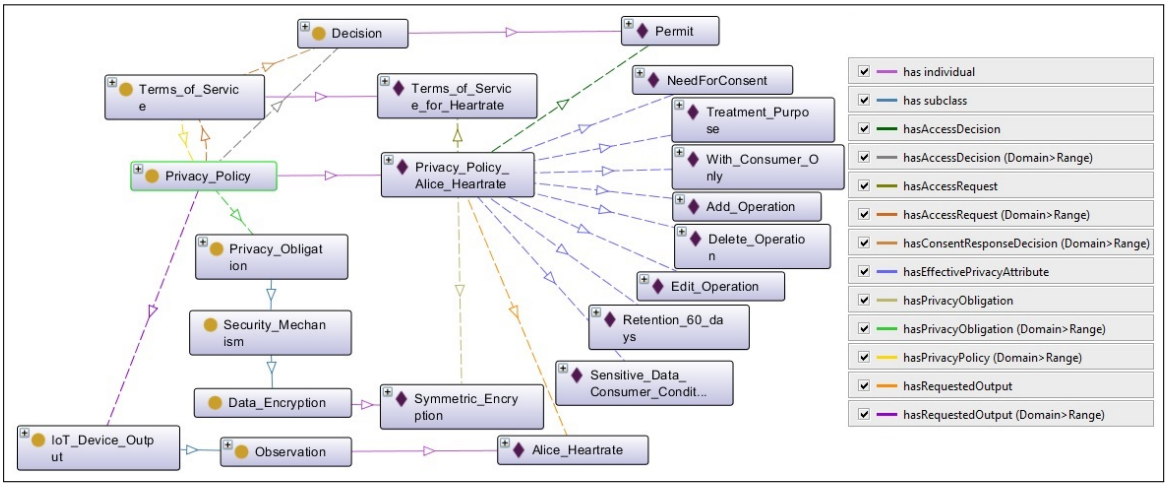


Figure 2.23: Example of a privacy policy instance

2.5 Summary

Ontology can be used to incorporate privacy legislation into privacy policies while considering several privacy requirements in the IoT domain. By using semantics, the data owners can create fine-grained privacy preferences for their data and the data consumers can define their terms of service by specifying how the shared data will be handled by their systems. In this chapter, we presented LIoPY, a European Legal compliant ontology for supporting preserving IoT PrivacY. LIoPY is defined over standardized concepts that are extended by appropriate privacy and security properties in order to be as interoprable as possible. Thus, LIoPY aims at defining a common privacy vocabulary using OWL, the standard ontology language to address the privacy requirements in the IoT environment. Moreover, LIoPY aims at making the IoT devices more autonomous by giving them the ability to deduce the data consumer's access rights according to the defined terms of service and the owner's privacy preferences.

After presenting our defined LIoPY's ontology and the privacy preferences model, we define in the next chapter a reasoning process by defining, implementing, and validating a privacy attribute matching algorithm.

Chapter 3

Semantic Rule Manager: Reasoning process validation

Table of Contents

3.1 Introduction 71

3.2 Reasoning process 72

3.2.1 Privacy Attribute Matching algorithm definition 72

3.2.2 Privacy Attribute Matching algorithm implementation 75

3.2.3 Privacy Attribute Matching algorithm validation: Semantic Rule Manager . 79

3.3 Experimentation 81

3.3.1 Motivating scenario 81

3.3.2 Experimental environment 81

3.3.3 Experimental results 82

3.4 Summary 86

3.1 Introduction

In the previous chapter, we proposed the LIoPY ontology and the theoretical privacy preferences model. In this chapter, we detail our reasoning process that is based on a set of defined inference rules in Section 3.2. Therefore, we first define a privacy attribute matching algorithm. Then, we implement it by defining a set of inference rules. After that, we validate it by introducing an overview of the proposed semantic rule manager architecture. Besides, we define an example of a LIoPY application to demonstrate its feasibility for supporting preserving privacy in the IoT domain in Section 3.3. Thus, we introduce a motivating scenario, present the experimental environment, and analyze the obtained results. Finally, we summarize the content of this chapter in Section 3.4.

3.2 Reasoning process

In this section, we illustrate our proposed algorithm definition and implementation. After that, we validate it by giving an overview of the semantic rule manager architecture.

3.2.1 Privacy Attribute Matching algorithm definition

Instantiating a privacy policy is the result of a successful privacy policy matching between the data owner's rules and the data consumer's terms of service using the defined privacy attributes. For this purpose, we define an algorithm, called Privacy Attribute Matching that is detailed on Algorithm 1. Several notations are used in Algorithm 1, thus Table 3.1 illustrates them with their descriptions. It is worth noting that all the used acronyms are detailed in the privacy preferences model (see Section 2.4, Chapter 2).

Table 3.1: Notations

Acronym	**Descriptions**
IoTDeviceOutputs	A set of instances of the IoT_Device_Output class
DCategory	An instance of the Data_Category class
ic	Intended Consent, An instance of the Consent class
crd	Consent Response Decision, An instance of the Decision class
ec	Effective Consent, An instance of the Consent class
intendedPA	Intended privacy attribute, A set of instances of the sub-classes of the Privacy_Attribute class
Anc_i	Ancestors of i, A set of instances of the same class as *i*
Des_i	Descendants of i, A set of instances of the same class as *i*
ip	Intended purpose, An instance of the Purpose class
$\mathcal{A}ip$	Allowed intended purpose, A set of instances of the Purpose class
$\mathcal{P}ip$	Prohibited intended purpose, A set of instances of the Purpose class
rp	Requested purpose, An instance of the Purpose class
ep	Effective purpose, An instance of the Purpose class
io	Intended operation, An instance of the Operation class
$\mathcal{A}io$	Allowed intended operation, A set of instances of the Operation class
$\mathcal{P}io$	Prohibited intended operation, A set of instances of the Operation class
ro	Requested operation, An instance of the Operation class
eo	Effective operation, An instance of the Operation class
idisc	Intended disclosure, An instance of the Disclosure class
$\mathcal{A}idisc$	Allowed intended disclosure, A set of instances of the Disclosure class
$\mathcal{P}idisc$	Prohibited intended disclosure, A set of instances of the Disclosure class
rdisc	Requested disclosure, An instance of the Disclosure class
edisc	Effective disclosure, An instance of the Disclosure class
ir	Intended retention, An instance of the Retention class
rr	Requested retention, An instance of the Retention class
er	Effective retention, An instance of the Retention class
icond	Intended condition, An instance of the Condition class
econd	Effective condition, An instance of the Condition class

Algorithm 1 takes as input the data consumer's terms of service and returns an instantiated privacy policy if there is a match with the appropriate privacy rules of a specific data owner. First, the algorithm checks if the related IoT devices collect the requested data using its set of device outputs $IoTDeviceOutputs$ (lines 4-9). If no, it returns an empty privacy policy (lines 10-12). Otherwise, it retrieves the appropriate data category's privacy rule. Then, it matches the data owner's privacy rule and the consumer's terms of service to instantiate a privacy policy (lines 13-60). According to each privacy attribute, a matching type is defined. For instance, when an access to an IoT device output is requested, the requested disclosure $ToS.rdisc$ is checked against the intended allowed disclosures A_{idisc} and the intended prohibited disclosures P_{idisc} according to the privacy rule of the output category $PRule$ (lines 34-42). Each privacy attribute compliance is already detailed on section 2.4.2.1, Chapter 2. When all the privacy rule's privacy attributes match all the privacy attributes of the terms of service, the privacy policy inherits all the requested privacy attributes. Moreover, the privacy policy inherits the privacy obligations associated with the considered privacy rule (line 64). Thus, the generated privacy policy defines how the data can be handled by the data consumer once shared.

Algorithm 1: Privacy policy generation through privacy requirement matching.

Input: *ToS* ▷ Instance of the *Terms_of_Service* class
Output: *PPolicy* ▷ Instance of the *Privacy_Policy* class

Function Privacy Attribute Matching (*ToS*):
  PRule ← ∅ ▷ Instance of the *Privacy_Rule* class
  PPolicy ← ∅
  foreach *(output in IoTDeviceOutputs)* **do**
    if *(output==ToS.requested_data)* **then**
      DCategory = output.hasDataCategory
      PRule = DCategory.hasPrivacyRule
    end
  end
  if *(PRule == ∅)* **then**
    return *PPolicy*
  end
  if *PRule.ic==NoNeedForConsent* ***or*** *(PRule.ic==NeedForConsent* ***and*** *ToS.crd==Permit)* **then**
    foreach *(ipa in intendedPA)* **do**
      switch *ipa* **do**
        case *Purpose* **do**
          if $(ToS.rp \notin (\cup_{pip_k \in Pip} Anc_{PRule.ip_k}) \cup (\cup_{pip_k \in Pip} Des_{PRule.ip_k})$ ***and*** $(ToS.rp \in \cup_{aip_j \in Aip} Des_{PRule.ip_j})$ **then**
            PPolicy.ep = ToS.rp
          end
          else
            return *PPolicy*
          end
        end
        case *Operation* **do**
          if $(ToS.ro \notin (\cup_{pio_k \in Pio} Anc_{PRule.io_k}) \cup (\cup_{pio_k \in Pio} Des_{PRule.io_k})$ ***and*** $(ToS.ro \in \cup_{aio_j \in Aio} Des_{PRule.io_j})$ **then**
            PPolicy.eo = ToS.ro
          end
          else
            return *PPolicy*
          end
        end
        case *Disclosure* **do**
          if $(ToS.rdisc \notin (\cup_{pidisc_k \in Pidisc} Anc_{PRule.idisc_k}) \cup (\cup_{pidisc_k \in Pidisc} Des_{PRule.idisc_k})$ ***and*** $(ToS.rdisc \in \cup_{aidisc_j \in Aidisc} Des_{PRule.idisc_j})$ **then**
            PPolicy.edisc = ToS.rdisc
          end
          else
            return *PPolicy*
          end
        end
        case *Retention* **do**
          if *(PRule.ir < ToS.rr)* **then**
            return *PPolicy*
          end
          else
            PPolicy.er = ToS.rr
          end
        end
        case *Condition* **do**
          if *(PRule.icond.allowedRole != ToS.hasInitiator.hasRole)* **then**
            return *PPolicy*
          end
          else
            PPolicy.econd = PRule.icond
          end
        end
      end
    end
    PPolicy.ec = PRule.ic
    PPolicy.hasRequestedOutput= output
    PPolicy.hasAccessDecision= Permit
    PPolicy.hasPrivacyObligation = PRule.hasPrivacyRuleObligation
  end
  return *PPolicy*
End Function

After defining the Privacy Attribute Matching Algorithm, we present in the following section the implementation of the proposed algorithm.

3.2.2 Privacy Attribute Matching algorithm implementation

Although the OWL language is used to implement LIoPY's classes, the defined algorithm cannot be expressed in OWL. For this reason, we use the Semantic Web Rule Language (SWRL) to define a set of inference rules, which are built upon different LIoPY's concepts and properties. The inference rule represents a set of conjunctions of atoms, called an antecedent that implies a result, called a consequent. Thus, based on SWRL rules and LIoPY's classes multiple privacy policies can be inferred for different possible data sharing cases in the real world. We propose below the set of inference rules.

Before applying the principle inference rules that match and derive a new privacy policy, some pre-processing inference rules need to be applied in order to speed up the privacy policy derivation's computing time.

For instance, the ancestors and the descendants of the three privacy attributes, namely Purpose, Operation, and Disclosure are computed using the following six inference rules.

LIoPY:Purpose(?p) ∧ LIoPY:hasPurpose-Ancestor(?p, ?a) ∧ LIoPY:hasPurpose-Ancestor(?a, ?aa) ⟶ LIoPY:hasPurpose-Ancestor(?p, ?aa)	LIoPY:Purpose(?p) ∧ LIoPY:hasPurpose-Descendant(?p, ?d) ∧ LIoPY:hasPurpose-Descendant(?d, ?dd) ⟶ LIoPY:hasPurpose-Descendant(?p, ?dd)
LIoPY:Operation(?op) ∧ LIoPY:hasOperation-Ancestor(?op, ?a) ∧ LIoPY:hasOperation-Ancestor(?a, ?aa) ⟶ LIoPY:hasOperation-Ancestor(?op, ?aa)	LIoPY:Operation(?op) ∧ LIoPY:hasOperation-Descendant(?op, ?d) ∧ LIoPY:hasOperation-Descendant(?d, ?dd) ⟶ LIoPY:hasOperation-Descendant(?op, ?dd)
LIoPY:Disclosure(?dis) ∧ LIoPY:hasDisclosure-Ancestor(?dis, ?a) ∧ LIoPY:hasDisclosure-Ancestor(?dis, ?aa) ⟶ LIoPY:hasDisclosure-Ancestor(?dis, ?aa)	LIoPY:Disclosure(?dis) ∧ LIoPY:hasDisclosure-Descendant(?dis, ?d) ∧ LIoPY:hasDisclosure-Descendant(?dis, ?dd) ⟶ LIoPY:hasDisclosure-Descendant(?dis, ?dd)

By using LIoPY, the data owner can define the data sensitivity and the preferences about how the shared data must be handled by consumers. However, the owner generally has not enough expertise in the privacy domain to choose the appropriate privacy obligations. For this purpose, we based our work on the existing privacy legislation [GDPR, 2016] and standards [ISO/IEC29100, 2011][OECD, 1981] in order to propose a set of inference rules that defines the privacy obligations related to the data category's privacy rule.

Illustration 1: sensitive data are rich in owner-specific habits. For this reason, we define the following inference rule that once applied, the reasoner will add the Symmetric_Encryption security obligation to the privacy rule that is associated to the data category, named Sensitive_Category:

```
LIoPY:Data_Category(?datCat) ∧ sameAs(?datCat, LIoPY:Sensitive_Category) ∧
LIoPY:Privacy_Rule(?privRule) ∧ LIoPY:hasPrivacyRule(?datCat, ?privRule) ⟶
LIoPY:hasPrivacyRuleObligation(?privRule, LIoPY:Symmetric_Encryption)
```

Illustration 2: data must be encrypted if the processing purpose is the first use purpose [GDPR, 2016]. For instance, we define the following SWRL rule that imposes applying the Symmetric_Encryption mechanism on the shared data when the data purpose is First_Use_Purpose:

```
LIoPY:Privacy_Rule(?privRule) ∧
LIoPY:hasAllowedIntendedPrivacyAttribute(?privRule, LIoPY:First_Use_Purpose) ⟶
LIoPY:hasPrivacyRuleObligation(?privRule, LIoPY:Symmetric_Encryption)
```

Illustration 3: data must be anonymized if the processing purpose is the second use purpose [GDPR, 2016]. For instance, we define the following SWRL rule that imposes applying the l-diversity mechanism [Machanavajjhala et al., 2006] on the shared data when the data purpose is Second_Use_Purpose:

```
LIoPY:Privacy_Rule(?privRule) ∧
LIoPY:hasAllowedIntendedPrivacyAttribute(?privRule, LIoPY:Second_Use_Purpose) ⟶
LIoPY:hasPrivacyRuleObligation(?privRule, LIoPY:l-diversity)
```

Illustration 4: data must be noised if the disclosure type is public [GDPR, 2016]. For instance, we define the following SWRL rule that imposes applying the Differential_Privacy mechanism [Dwork, 2008] on the shared data when the disclosure type is With_Public_Third_Parties:

```
LIoPY:Privacy_Rule(?privRule) ∧
LIoPY:hasAllowedIntendedPrivacyAttribute(?privRule, LIoPY:With_Public_Third_Parties) ⟶
LIoPY:hasPrivacyRuleObligation(?privRule, LIoPY:Differential_Privacy)
```

Besides the privacy obligation inference rules, we define two inference rules to handle the data owner consent. Thus, when the privacy rule imposes explicit data owner's consent to be applied, the data Owner should be contacted to request the own consent. The Consumer has a property hasReceiverConsentRequest whose value will be inferred using the following SWRL rule:

```
LIoPY:Owner(?owner) ∧ LIoPY:owns(?owner, ?dev) ∧ sosa:madeObservation(?dev, ?d) ∧
sosa:Observation(?d) ∧ LIoPY:observation_name(?d, ?dname) ∧ LIoPY:Terms_of_Service(?tos)
∧ LIoPY:requested_data_name(?tos, ?rname) ∧ swrlb:stringEquallgnoreCase(?rname, ?dname) ∧
LIoPY:hasDataCategory(?d, ?dc) ∧ LIoPY:hasPrivacyRule(?dc, ?pr) ∧
LIoPY:hasAllowedIntendedPrivacyAttribute(?pr, ?ipa) ∧ sameAs(?ipa, LIoPY:NeedForConsent)
⟶ LIoPY:hasReceiverConsentRequest(?tos, ?owner)
```

When a consent request is received, the data owner decides to accept or reject the request by defining the value of the *hasConsentResponseDecision* object property. Both *Permit* and *Deny* are the possible values, as specified on the following SWRL rules:

LIoPY:Owner(?owner) ∧ LIoPY:Terms_of_Service(?tos) ∧ LIoPY:hasReceiverConsentRequest(?tos, ?owner) ⟶ LIoPY:hasConsentResponseDecision(?tos, LIoPY:Permit)	LIoPY:Owner(?owner) ∧ LIoPY:Terms_of_Service(?tos) ∧ LIoPY:hasReceiverConsentRequest(?tos, ?owner) ⟶ LIoPY:hasConsentResponseDecision(?tos, LIoPY:Deny)

Once the previous inference rules are applied, the privacy policy derivation rule can be used. Figure 3.1 shows an example of an inference rule that instantiates a privacy policy. This inference rule considers the case of no explicit data owner's consent is required.

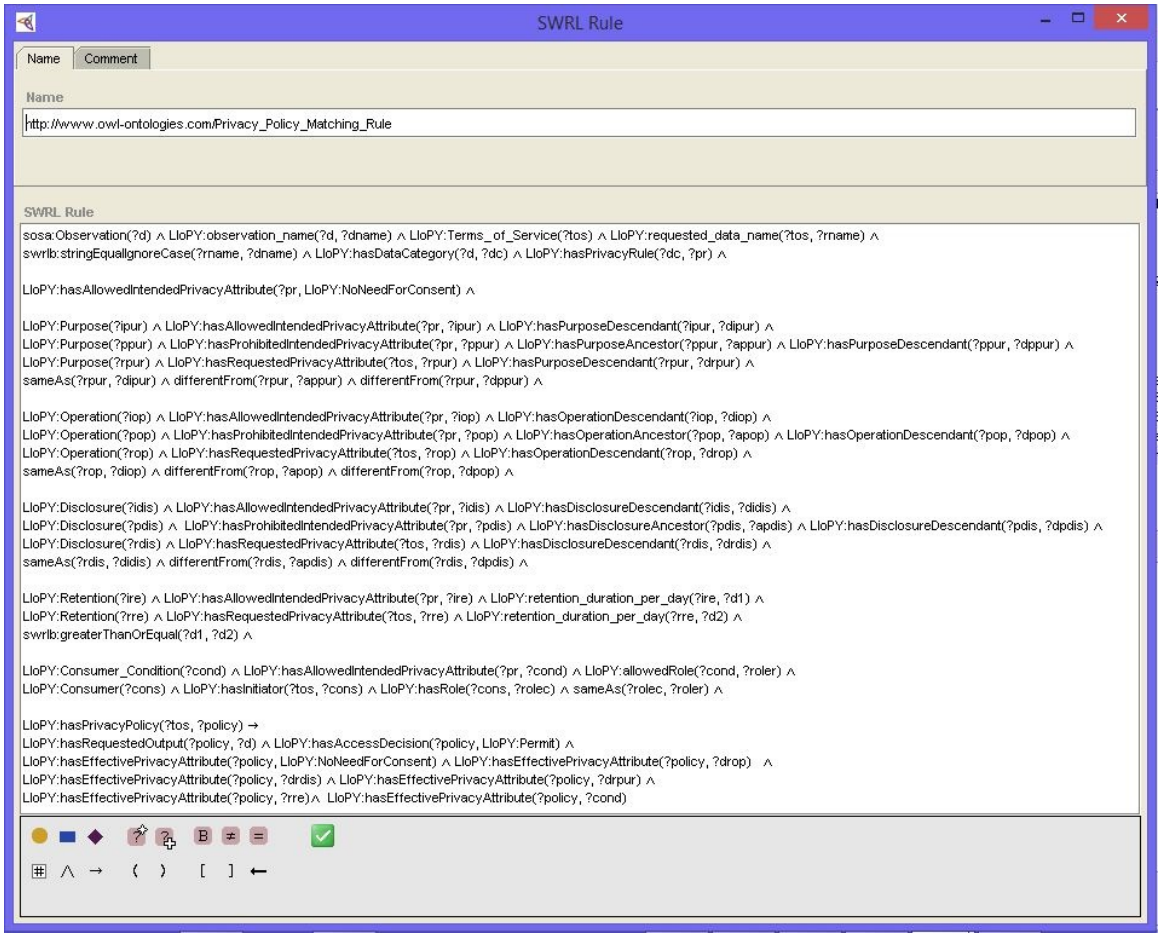


Figure 3.1: Example of a privacy matching SWRL rule for instantiating a Privacy_Policy

Another example of an inference rule is listed below that instantiates a privacy policy in the case of a match between the terms of service and the privacy rule of the requested IoT data category. We define a matching type according to each privacy attribute. For instance, when an access to an observation is requested, the requested purpose is checked against the intended allowed and prohibited purposes according to the privacy rule of the observation category. In case of a requested purpose compliance, the derived privacy policy inherits the requested purpose's descendants ($LIoPY : hasEffectivePrivacyAttribute(?policy, ?drpur)$).

sosa:Observation(?d) ∧ LIoPY:observation_name(?d, ?dname) ∧ LIoPY:Terms_of_Service(?tos) ∧ LIoPY:requested_data_name(?tos, ?rname) ∧ swrlb:stringEqualIgnoreCase(?rname, ?dname) ∧ LIoPY:hasDataCategory(?d, ?dc) ∧ LIoPY:hasPrivacyRule(?dc, ?pr) ∧ LIoPY:hasAllowedIntendedPrivacyAttribute(?pr, LIoPY:NeedForConsent) ∧ LIoPY:hasConsentResponseDecision(?tos, LIoPY:Permit) ∧ LIoPY:Purpose(?ipur) ∧ LIoPY:hasAllowedIntendedPrivacyAttribute(?pr, ?ipur) ∧ LIoPY:hasPurposeDescendant(?ipur, ?dipur) ∧ LIoPY:Purpose(?ppur) ∧ LIoPY:hasProhibitedIntendedPrivacyAttribute(?pr, ?ppur) ∧ LIoPY:hasPurposeAncestor(?ppur, ?appur) ∧ LIoPY:hasPurposeDescendant(?ppur, ?dppur) ∧ LIoPY:Purpose(?rpur) ∧ LIoPY:hasRequestedPrivacyAttribute(?tos, ?rpur) ∧ LIoPY:hasPurposeDescendant(?rpur, ?drpur) ∧ sameAs(?rpur, ?dipur) ∧ differentFrom(?rpur, ?appur) ∧ differentFrom(?rpur, ?dppur) ∧ LIoPY:Operation(?iop) ∧ LIoPY:hasAllowedIntendedPrivacyAttribute(?pr, ?iop) ∧ LIoPY:hasOperationDescendant(?iop, ?diop) ∧ LIoPY:Operation(?pop) ∧ LIoPY:hasProhibitedIntendedPrivacyAttribute(?pr, ?pop) ∧ LIoPY:hasOperationAncestor(?pop, ?apop) ∧ LIoPY:hasOperationDescendant(?pop, ?dpop) ∧ LIoPY:Operation(?rop) ∧ LIoPY:hasRequestedPrivacyAttribute(?tos, ?rop) ∧ LIoPY:hasOperationDescendant(?rop, ?drop) ∧ sameAs(?rop, ?diop) ∧ differentFrom(?rop, ?apop) ∧ differentFrom(?rop, ?dpop) ∧ LIoPY:Disclosure(?idis) ∧ LIoPY:hasAllowedIntendedPrivacyAttribute(?pr, ?idis) ∧ LIoPY:hasDisclosureDescendant(?idis, ?didis) ∧ LIoPY:Disclosure(?pdis) ∧ LIoPY:hasProhibitedIntendedPrivacyAttribute(?pr, ?pdis) ∧ LIoPY:hasDisclosureAncestor(?pdis, ?apdis) ∧ LIoPY:hasDisclosureDescendant(?pdis, ?dpdis) ∧ LIoPY:Disclosure(?rdis) ∧ LIoPY:hasRequestedPrivacyAttribute(?tos, ?rdis) ∧ LIoPY:hasDisclosureDescendant(?rdis, ?drdis) ∧ sameAs(?rdis, ?didis) ∧ differentFrom(?rdis, ?apdis) ∧ differentFrom(?rdis, ?dpdis) ∧ LIoPY:Retention(?ire) ∧ LIoPY:hasAllowedIntendedPrivacyAttribute(?pr, ?ire) ∧ LIoPY:retention_duration_per_day(?ire, ?d1) ∧ LIoPY:Retention(?rre) ∧ LIoPY:hasRequestedPrivacyAttribute(?tos, ?rre) ∧ LIoPY:retention_duration_per_day(?rre, ?d2) ∧ swrlb:greaterThanOrEqual(?d1, ?d2) ∧ LIoPY:Consumer_Condition(?cond) ∧ LIoPY:hasAllowedIntendedPrivacyAttribute(?pr, ?cond) ∧ LIoPY:allowedRole(?cond, ?roler) ∧ LIoPY:Consumer(?cons) ∧ LIoPY:hasInitiator(?tos, ?cons) ∧ LIoPY:hasRole(?cons, ?rolec) ∧ sameAs(?rolec, ?roler) ∧ LIoPY:hasPrivacyPolicy(?tos, ?policy) ⟶ LIoPY:hasRequestedOutput(?policy, ?d) ∧ LIoPY:hasAccessDecision(?policy, LIoPY:Permit) ∧ LIoPY:hasEffectivePrivacyAttribute(?policy, LIoPY:NeedForConsent) ∧ LIoPY:hasEffectivePrivacyAttribute(?policy, ?drop) ∧ LIoPY:hasEffectivePrivacyAttribute(?policy, ?drdis) ∧ LIoPY:hasEffectivePrivacyAttribute(?policy, ?drpur) ∧ LIoPY:hasEffectivePrivacyAttribute(?policy, ?rre)∧ LIoPY:hasEffectivePrivacyAttribute(?policy, ?cond)

After instantiating the privacy policy, some obligations that should be applied to the data before sharing them with third-parties are added to each instantiated privacy policy. To this end, we define the following SWRL rule in order to infer the privacy obligation according to the data category's privacy rule of the requested data.

LIoPY:Privacy_Policy(?priPol) ∧ LIoPY:hasAccessDecision(?priPol, LIoPY:Permit) ∧
LIoPY:hasRequestedOutput(?priPol, ?reqOut) ∧ LIoPY:hasDataCategory(?reqOut, ?datCat) ∧
LIoPY:Privacy_Rule(?privRule) ∧ LIoPY:hasPrivacyRule(?datCat, ?privRule) ∧
LIoPY:hasPrivacyRuleObligation(?privRule, ?oblig) ⟶
LIoPY:hasPrivacyObligation(?priPol, ?oblig)

Once the privacy policy is instantiated and the privacy obligations are added, we need to ensure the policy enforcement. To this end, we define the following inference rule that enforces the retention limitation principle. The following rule defines the condition that leads to deny a Privacy_Policy. In other words, the hasAccessDecision becomes Deny when the period between the current time and the take_effect_date of the privacy policy is greater than the retention duration.

LIoPY:Privacy_Policy(?priPol) ∧ LIoPY:hasAccessDecision(?priPol, LIoPY:Permit) ∧
LIoPY:take_effect_date(?priPol, ?ted) ∧ LIoPY:Retention(?r) ∧
LIoPY:hasEffectivePrivacyAttribute(?priPol, ?r) ∧ LIoPY:retention_duration_per_day(?r, ?rd) ∧
temporal:duration(?ret, "now", ?ted, "days") ∧ swrlb:greaterThan(?ret, ?rd) ⟶
LIoPY:hasAccessDecision(?priPol, LIoPY:Deny)

Thus, LIoPY grants the data owner the rights to control the own data and prevents malicious parties from violating the IoT data privacy.

After defining and implementing the Privacy Attribute Matching Algorithm, we introduce in the following section the validation of the proposed algorithm.

3.2.3 Privacy Attribute Matching algorithm validation: Semantic Rule Manager

In order to validate the proposed algorithm, we propose a Semantic Rule Manager, which includes both the LIoPY ontology and the defined set of inference rules. The Semantic Rule Manager aims at matching the data owner's privacy preferences and the data consumer's terms of service in order to generate an adapted privacy policy. The Semantic Rule Manager allows the data owners to specify their privacy preferences. This manager can then evaluate the received data consumer's terms of service taken as an input in order to generate a common privacy policy returned as an output only in case of a match between the data consumer's privacy promises and the data owner's privacy preferences.

Figure 3.2 shows the architecture of the Semantic Rule Manager. The architecture includes five core components, which are (i) inference rules, which are the rules that enable matching the terms of service and the privacy rules, (ii) privacy preferences, which are the associated privacy rules to the data owner's data, (iii) rule engine, which is responsible for reasoning about the received terms of service and, then tacking a decision to create or not a privacy policy, (iv) query engine, which enables the rule engine to interrogate the LIoPY ontology, and (v) LIoPY ontology, which includes the different concepts and properties introduced in the previous chapter (see Chapter 2).

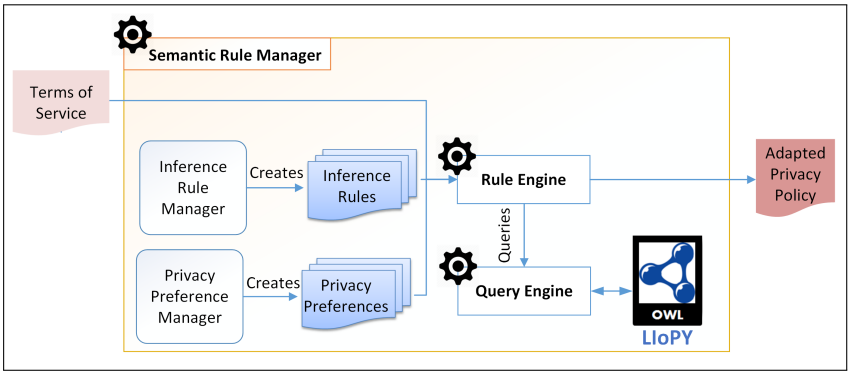


Figure 3.2: Architecture of Semantic Rule Manager

The Semantic Rule Manager is deployed on the data owner's gateway that can be, for example, a tablet computer. It is a component of the proposed privacy policy sharing process depicted on Figure 3.3. Through the owned gateway, the data owner uses (i) the Privacy Preference Manager to define preferences about each device output category and (ii) the MQTT client to create a "Produced IoT data Topic" and associate the appropriate data producer that will publish its collected IoT data on the created topic. The data owner can define two types of preferences, namely permission settings and privacy rules. The data consumer publishes its terms of service after creating its "Terms of service Topic". The data producer subscribes on a data consumer's topic in order to benefit from the data consumer's service and receive the consumer's terms of service updates.

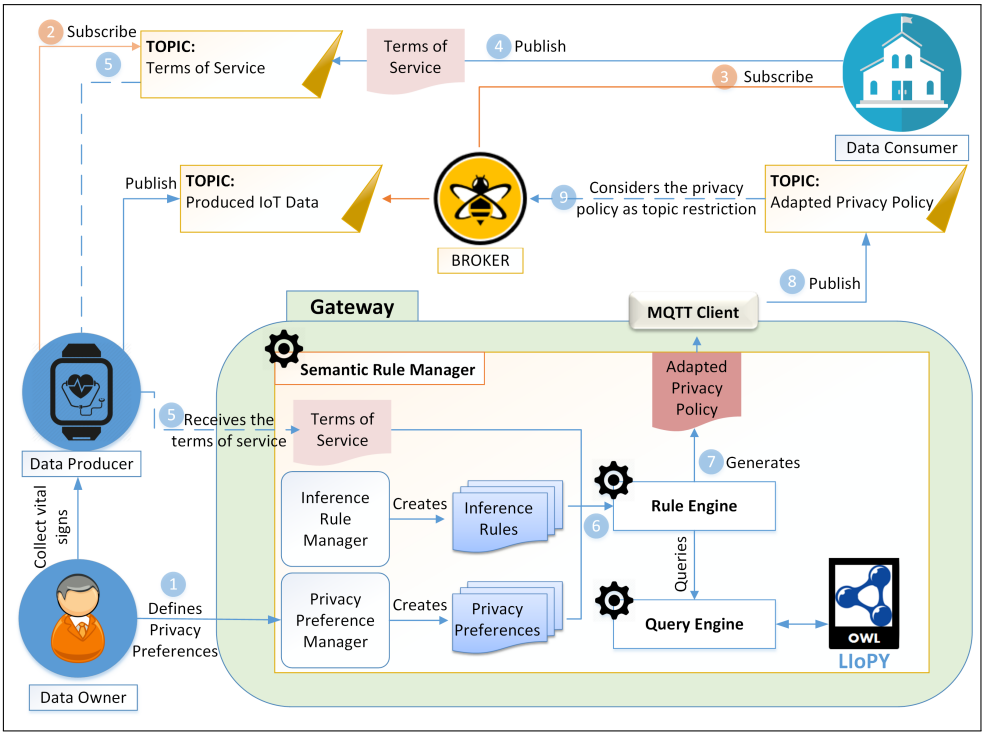


Figure 3.3: Privacy policy sharing process

When new terms of service are received, the data producer (i.e., IoT device) communicates with the Semantic Rule Manager, which is responsible for reasoning about the received terms of service and, then tacking a decision to create or not a privacy policy. During the reasoning process, the rule engine evaluates the terms of service by using the predefined set of inference rules. In case of a match between the data consumer's privacy promises and the data owner's privacy preferences, a privacy policy is generated and sent to the gateway that published this privacy policy on the "Adapted Privacy Policy Topic". This privacy policy is considered as data producer's topic restrictions that permit the data consumer's subscription. Thus, the data consumer receives the collected IoT data by the data producer only if it is allowed in the privacy policy.

3.3 Experimentation

In this section, we first define a motivating scenario, then introduce the chosen experimental environment, and finally present the obtained experimental results in terms of feasibility and performance.

3.3.1 Motivating scenario

As said before, we illustrate our ideas in the healthcare context, but the LIoPY ontology is application agnostic and can be applied in other IoT contexts. Healthcare is one of the challenging domain where privacy needs to be addressed in order to prevent disclosing patient's medical data. In this context, we describe bellow a motivating healthcare scenario:

Alice, a 40-year old woman who suffers from a heart disease. Preferred to stay at home, she accepted to use a wireless body sensor that will continuously check her health conditions by measuring her heart rate and her position. The sensor collects these data and sends them to the home-gateway through a secure channel. From the medical center, Alice's doctor can remotely monitor her health by receiving Alice's heart rate every few minutes. During the treatment period, the doctor can access the data and add some remarks to Alice's results. In the case of a cardiac problem, the smart device alerts the emergency service by sending Alice's heart rate and position. Then, the hospital dispatches an ambulance to help her. The emergency service can disclose Alice's position to the traffic monitoring service in order to ask for the best route to Alice's location and save valuable time. In addition, the home-gateway enables Alice to adjust her device settings, including permission and access control. In fact, Alice can add additional people to be notified in case of emergency, such as some of her family members. However, Alice is afraid that one of the authorized people uses her device to monitor her position even in the absence of an emergency case.

3.3.2 Experimental environment

In order to validate the proposed reasoning process, we implement the defined privacy policy sharing process. For this purpose, we use Java programming language to develop the semantic

rule manager, which is based on Algorithm 1. It includes both the LIoPY ontology and the set of inference rules. It gets the terms of service as an input and generates a common privacy policy as an output. The reason behind the Java language choice is that this language enables ontology manipulation thanks to a set of Java API. In our case, we use both OWLAPI [Horridge and Bechhofer, 2011] and SWRLAPI [O'Connor et al., 2008]. The first API is an open source Java API and a reference implementation for creating, manipulating and serializing OWL Ontologies. While the second one is a Java API for working with the OWL-based SWRL rule and SQWRL query languages.

In order to reuse the developed semantic rule manager, we extract it as a Java ARchive (JAR) file, which is a package file format used to store many Java classes and associated metadata into one file for distribution. Indeed, the developed Java Archive can be easily integrated in other applications thanks to its included Java-specific manifest file specifying the entry point class to launch it.

3.3.3 Experimental results

With the experimental stage, we first want to proof the LIoPY feasibility, and second to measure its performance.

In the "Alice LIoPY instance" base, we can find Alice as an individual of the Owner class that owns a Heartrate_Sensor, which is an individual of the sosa:Sensor class. This sensor has a device output, called Alice_Heartrate. According to Alice, her heart rate is considered as sensitive information, thus it has the Sensitive_Data_Privacy_Rule as a privacy rule. In order to validate the reasoning process, let the Terms_of_Service_for_Heartrate an example of terms of service instance. Both Sensitive_Data_Privacy_Rule and Terms_of_Service_for_Heartrate are detailed on Figure 2.21 and Figure 2.22 in Chapter 2, respectively.

After launching a set of test sequence batteries, the appropriate privacy policies are derived and one of the obtained derivations is illustrated in Figure 3.4. It shows an example of a privacy policy, called Privacy_Policy_Alice_Heartrate that is generated after matching the Sensitive_Data_Privacy_Rule and the Terms_of_Service_for_Heartrate. Eight privacy attributes are generated as the range of the object property hasEffectivePrivacyAttribute. Moreover, the privacy policy inherits the Symmetric_Encryption obligation as expected.

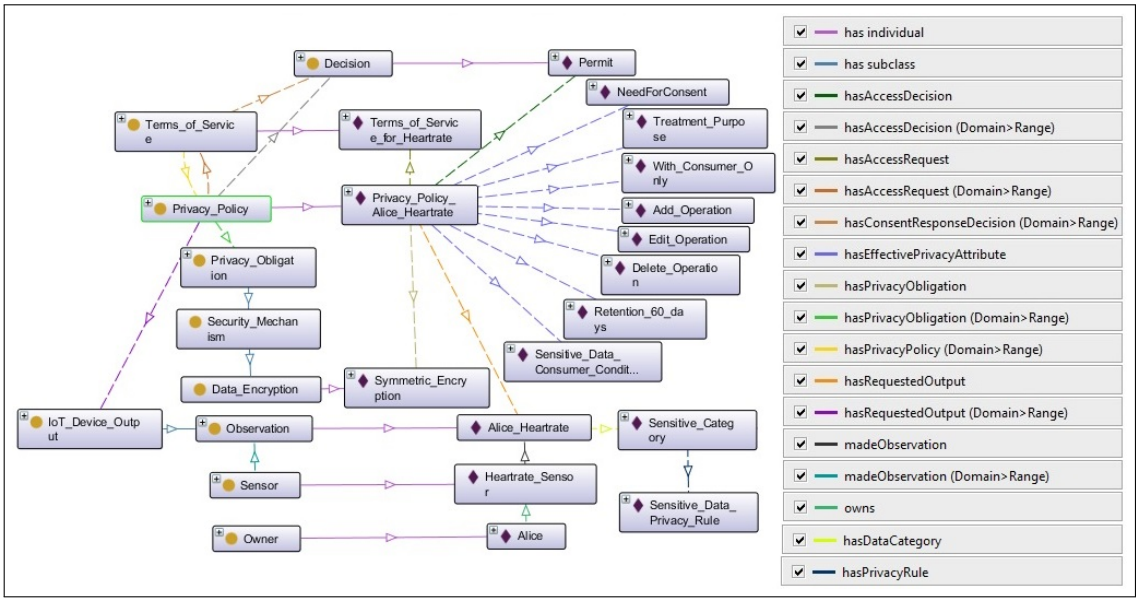


Figure 3.4: After privacy policy derivation

For each test, we consider a different instance of the Privacy_Rule class and the same instance of the Terms_of_Service class in order to check several possibilities. The obtained result proves that the reasoning process succeeds at inferring a privacy policy in case of a match between the privacy rules and the terms of service.

After proving the LIoPY feasibility, we conduct some experiments to measure the performance of the semantic rule manager. To this end, we intend to evaluate if the computing time of reasoning is acceptable by making several tests while increasing the number of terms of service individuals from 1 to 20. Hence, we perform an experiment to measure the required time to check and create the privacy policy for monitoring vital sign for normal condition. The number of terms of service individuals increases from 1 to 20. The response time is equal to the matching time plus the LIoPY's upload and update.

Figure 3.5 shows the processing time of the privacy policy derivation. We observe that the semantic rule manager can support a large number of individuals within reasonable processing time. Indeed, the response time varies from 14 to 100 seconds. The response time includes the matching time, which is the required time to compute a new privacy policy in case of a match. We deduce that the rise of the matching time is less than the response time rise. Indeed, the difference between having 1 individual and 20 individuals in terms of response time is around 88 seconds, while the difference between having 1 individual and 20 individuals in terms of matching time is around 35 seconds. This lets us conclude that the increase of individuals dœs not affect the matching performance of the semantic rule manager. Thus, LIoPY's upload and update operations increase the response time. This can be addressed by optimizing the ontology management code. Moreover, the linearity property behind these results means that a better computer system setting would obtain a lower processing time.

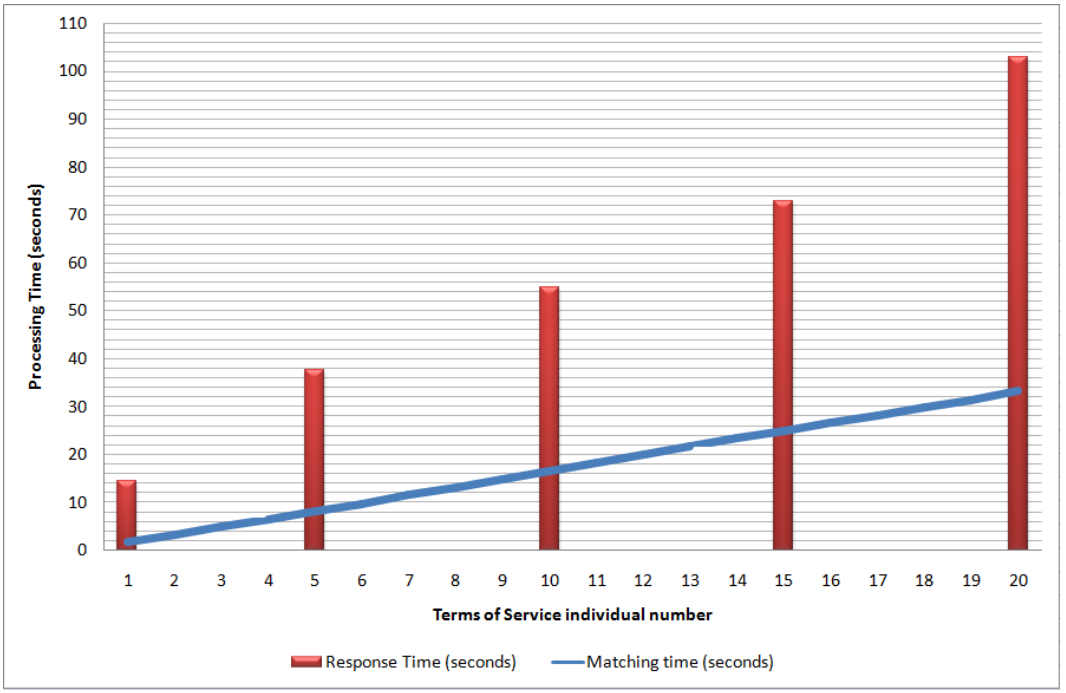


Figure 3.5: Processing time of privacy policy derivation

After measuring the performance time, we conduct a new experiment in order to analyze the required computational resources for the reasoning process in terms of CPU usage and used memory size. Thus, we run the semantic rule manager in order to match fifty terms of service individuals with the defined privacy rules. The obtained results of this experiment are depicted on the following figures.

Figure 3.6 depicts the CPU usage during four minutes while matching fifty terms of service individuals. In this figure, we measure the CPU time as a percentage of the CPU's capacity, which is called the CPU usage. We observe that the CPU usage is high at the beginning, then it decreases and varies in a more or less fifty percent of the CPU's capacity. We can deduce that by increasing the number of the terms of service individuals, the CPU usage is not raised. It is worth noting that the experiment is launched on a machine equipped with an Intel i5 6200u dual-core with 8GB memory. Thus, the obtained results can be better when using a local server with more computing capabilities.

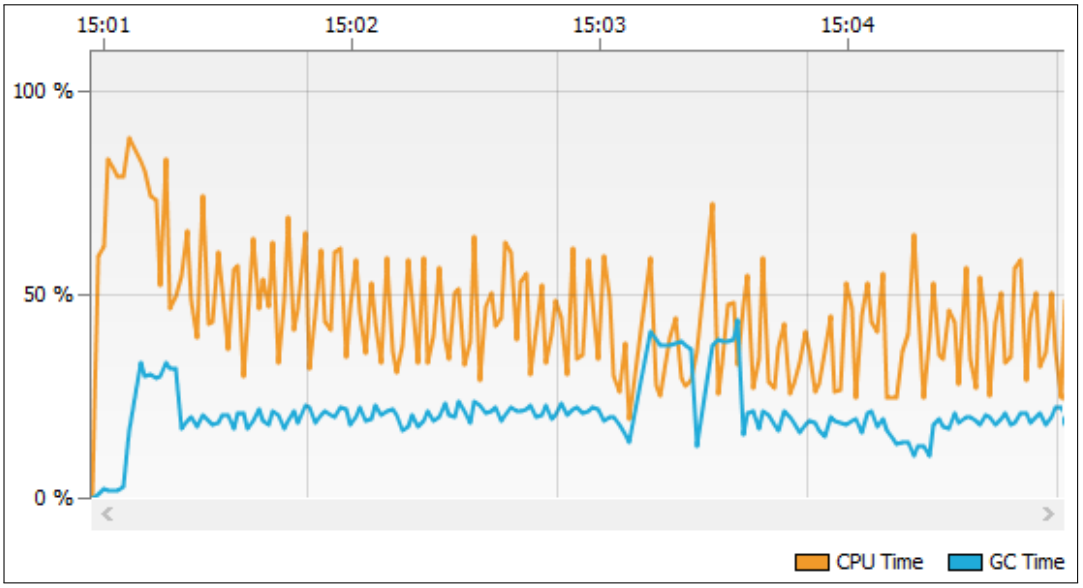


Figure 3.6: CPU usage

Figure 3.7 depicts a snapshot of the used memory size during four minutes while matching fifty terms of service individuals. The heap size is the amount of memory allocated to the semantic rule manager running in the Java Virtual Machine (JVM), while the used heap is the memory used by the semantic rule manager. In our case, the allocated memory dœs not exceed 2000 Megabytes and the used memory is lower than 1000 Megabytes. We can deduce that the required memory size is acceptable and can be integrated with a smart device in order to support our semantic-based solution.

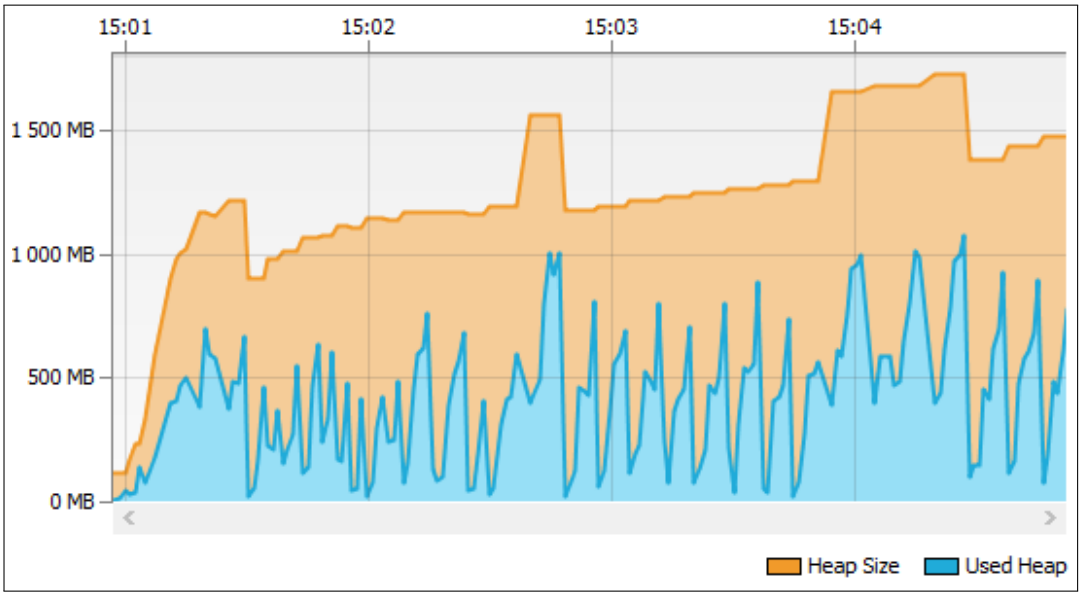


Figure 3.7: Used memory size

To sum up, the obtained results demonstrate the LIoPY capability to be instantiated in a real environment for preserving IoT privacy. Nevertheless, a gateway with high computation, memory, and storage capabilities is required to support the semantic rule manager.

3.4 Summary

IoT emergence presents an opportunity to improve efficiency and quality of life to the users. However, the analysis of the detailed data generated by the IoT devices raises the privacy risks. Semantic modeling is a used solution to give the data owner the control over the own data. However, the data owner has not enough experience and expertise in the privacy domain to take advantage of the legal rights. Thus, semantic modeling becomes fundamental to infer the required privacy obligations to preserve privacy. For these reasons, we have proposed a semantic rule manager based on a set of inference rules that enables new knowledge generation using the proposed LIoPY ontology. The semantic rule manager aims at generating a common privacy policy to address the privacy requirements in the IoT environment. Thus, SWRL is used to define the inference rules that implement the defined Privacy Attribute Matching algorithm. Our implementation is experimented in the healthcare context, but it can be applied in other IoT contexts.

As seen in the last two chapters, LIoPY and the inference rules can be used to express and match the privacy preferences of both the data owners and the data consumers. However, they cannot fully enforce the common privacy policy. In the next chapter, we introduce the next contribution, which is an end-to-end, distributed privacy-preserving framework for IoT data whose goal is to address the privacy requirement enforcement issue.

Chapter 4

PrivBlockchain: Blockchain-based IoT data privacy-preserving framework

Table of Contents

4.1 Introduction . . . 88

4.2 Background and Related Work . . . 88

4.2.1 Blockchain technology . . . 88

4.2.2 Homomorphic encryption technology . . . 90

4.3 Design goals . . . 91

4.4 PrivBlockchain: IoT data privacy-preserving framework . . . 92

4.4.1 PrivBlockchain overview . . . 92

4.4.2 PrivBlockchain core components . . . 95

4.4.3 PrivBlockchain modules . . . 99

4.5 Blockchain-based IoT device management module . . . 100

4.5.1 Smart contract description . . . 100

4.5.2 Privacy permission setting adding process . . . 101

4.6 Blockchain-based IoT data sharing module . . . 107

4.6.1 Smart contract description . . . 108

4.6.2 IoT data sharing process . . . 108

4.7 Homomorphic encryption-based IoT data aggregation module . . . 113

4.7.1 Smart contract description . . . 114

4.7.2 Privacy policy generation process . . . 114

4.7.3 IoT data aggregation process . . . 116

4.8 Summary . . . 120

4.1 Introduction

Internet of Things (IoT) has emerged as one of the most significant technology in different application domains, such as smart home, smart grid, and smart city. The IoT's benefit to individuals' lives is realized thanks to the analytics and aggregate information from the smart devices and the huge volumes of produced IoT data. Although multiple researchers have studied the privacy-preserving issue in the IoT domain, many challenges, such as the single point of failure, single point of trust, and raw data disclosure issues remain to be addressed. Motivated by these drawbacks, we focus on preserving the IoT data privacy during the whole IoT data lifecycle, from the owner's consent to the data analysis. Thus, the blockchain technology is used to address both the single point of failure and the single point of trust issues, while the raw data disclosure issue is tackled by the homomorphic encryption technology. Therefore, we first introduce some technical backgrounds used in this chapter, then we define our design goals for preserving IoT data privacy during all the lifecycle. Then, we overview our contribution in this chapter, which is an IoT data privacy-preserving framework, called PrivBlockchain that includes three modules based on semantic, blockchain, and homomorphic encryption technologies. After that, we detail each PrivBlockchain's module in a separate section.

This chapter is organized as follows. Background is given in Section 4.2. Section 4.3 defines our design goals. Section 4.4 presents our proposed IoT data privacy-preserving framework, called PrivBlockchain. The three PrivBlockchain's modules are detailed in Section 4.5, Section 4.6, and Section 4.7. Section 4.8 summarizes the content of this chapter.

4.2 Background and Related Work

As mentioned above, our contribution in this chapter is based on both blockchain and homomorphic encryption technologies, which are introduced in this section.

4.2.1 Blockchain technology

The blockchain technology is a distributed computing paradigm that successfully overcomes the problem related to the trust of a centralized party. Thus, in a blockchain network, several nodes collaborate among them to secure and maintain a set of shared transaction records in a distributed way without relying on any trusted party. Specific nodes in the network known as miners are responsible for collecting transactions into blocks, solving challenging computational puzzles in order to reach consensus, and adding the blocks to a distributed public ledger known as the blockchain.

The first proposed system based on this technology was Bitcoin [Nakamoto et al., 2008], which allows users to transfer securely the currency (bitcoins) without a centralized regulator. Besides, Ethereum [Buterin et al., 2014] is another blockchain-based system that can also be used for the cryptocurrency. Unlike Bitcoin, Ethereum has the ability to use a smart contract, which is a common agreement between two or more parties. It stores information, processes inputs, and

writes outputs thanks to its predefined functions [Buterin et al., 2014]. For instance, the smart contract can define the constructor function that enables the smart contract creation. Hosting a new smart contract on the blockchain is enabled by invoking the constructor function through a transaction, whose sender becomes the smart contract owner. A self-destruct function is another example of the functions that can be defined in a smart contract. Usually, only the smart contract owner can destruct the contract by invoking this function.
A smart contract is likely to be a class that contains state variables, functions, function modifiers, events, and structures [Buterin et al., 2014]. Besides, it can even call other smart contracts. We represent the smart contract, which is denoted as SC, as a tuple that has the following form:

$$SC =< states, functions >$$

- **States**: they are variables that hold some data or the owner's Ethereum wallet address (i.e., the address in which the smart contract is deployed). We can distinguish between two state types, namely *constant states*, which can never be changed, and *writable states*, which save states in the blockchain.

- **Functions**: they are pieces of code that can read or modify states. We can distinguish between two function types, namely *read-only functions*, which are marked as constant in the code and do not require *gas* [1] to run and *write functions* that require *gas* because the state transitions must be encoded in a new block of the blockchain. Furthermore, Ethereum requires paying currency to avoid infinitely runs of a smart contract.

Recently, other projects demonstrate how these blockchains can serve in other domains, such as the Storj project [Storj, 2014], which is a decentralized peer-to-peer cloud storage network, and the Onename project [Onename, 2016], which is a distributed and secured identity platform. Moreover, blockchain technology is also used in order to address the privacy issue in the IoT domain. However, the existing blockchain-based solutions [Ouaddah et al., 2016] [Zyskind et al., 2015] concentrate on addressing the access control issue in the IoT applications. In fact, they adapt the blockchain by eliminating financial bitcoin and introducing new types of transactions in order to limit unauthorized access. Other blockchain-based solutions [Biswas and Muthukkumarasamy, 2016] [Hashemi et al., 2016] [Dorri et al., 2017a] assumed that the IoT devices had sufficient resources to solve the Proof-Of-Work, which may not always be true. Moreover, the examples cited above did not generally address the whole data lifecycle. Besides, existing solutions did not consider all the privacy requirements, such as the purpose, retention duration, disclosure limitation, etc. that are defined by the privacy standard [ISO/IEC29100, 2011] and legislation [GDPR, 2016] to preserve the user's privacy. Furthermore, other researchers [Guan et al., 2018] [Wang et al., 2018] used the blockchain technology in order to protect the user's privacy during a data aggregation process in a smart grid. However, they used less sophisticated consensus mechanisms in order to eliminate the expensive proof-of-work solving computation.

[1] gas: a unit that measures the amount of computational effort that it will take to execute certain operations.

4.2.2 Homomorphic encryption technology

The homomorphic encryption technology allows privacy-preserving computation over encrypted data. The homomorphic encryption is a special encryption schema, in which some computation results can be obtained over ciphertext calculation without knowing the appropriate plaintexts and private keys of the ciphertexts [Acar et al., 2018]. Thus, an encryption scheme is called homomorphic over an algebraic operation, denoted as $\oplus$ only if $E(M_1 \oplus M_2)$ can be computed from $E(M_1) \oplus E(M_2)$, with $E()$ is a homomorphic encryption function and $M_1, M_2 \in Z_N$ are two data items. There are several homomorphic encryption schemes in the literature, such as RSA [Rivest et al., 1978], ElGamal [ElGamal, 1985], and Paillier cryptosystem [Paillier, 1999]. According to [Acar et al., 2018], both RSA and ElGamal cryptosystems are only multiplicatively homomorphic. Hence, they do not allow the homomorphic addition of ciphertexts. However, the Paillier cryptosystem implements the additive and multiplication operations. For this reason, we employ the Paillier cryptosystem in our study in order to address the privacy issue in the blockchain-based data aggregation process.
Principles of the Paillier cryptosystem are as follows [Acar et al., 2018]:

i. **Key generation.** Let $N = pq$, where p and q are two large primes such that $gcd(pq, (p-1)(q-1))$=1, with gcd represents the greatest common divisor. Let $\lambda = lcm(p-1, q-1)$, where lcm refers to the least common multiple. Then select $g \in Z^*_{N^2}$, such that N satisfies the order divisible by g. Set function $L(u)$ as $L(u) = (u-1)/N$ and check the existence of $\mu = L(g^\lambda \mod N^2))^{-1} \mod N$ to ensure N divides the order of g. Then, the public and private keys are generated as $Pk_{Pai} = (N, g)$ and $Sk_{Pai} = (\lambda(N), \mu)$, respectively.

ii. **Public key encryption.** For each message $M \in Z_N$, the number $r \in Z^*_N$ is randomly chosen and M is encrypted as follows:

$$C = Enc(M, Pk_{Pai}) = g^M r^N \mod N^2 \tag{4.1}$$

iii. **Private key decryption.** Let a ciphertext $C \in Z^*_{N^2}$, the decryption is done by:

$$M = Dec(C, Sk_{Pai}) = \frac{L(C^\lambda \mod N^2)}{L(g^\lambda \mod N^2)} \mod N \tag{4.2}$$

iv. **Evaluation.** Let $Enc(M_1, Pk_{Pai})$ and $Enc(M_2, Pk_{Pai})$ are two encrypted messages and $M_1, M_2 \in Z_N$, the ciphertext is calculated as follows:

$$\begin{aligned} Enc(M_1, Pk_{Pai}).Enc(M_2, Pk_{Pai}) &= g^{M_1} r_1^N g^{M_2} r_2^N \mod N^2 \\ &= g^{M_1+M_2} (r_1 r_2)^N \mod N^2 \\ &= Enc(M_1 + M_2, Pk_{Pai}) \end{aligned} \tag{4.3}$$

As a result, without knowing M_1 and M_2 plaintexts, the encrypted value of $M_1 + M_2$ can be obtained. The private key holder reads the plaintext sum by using the decryption function.

After introducing some technical backgrounds used in this chapter, we define our design goals, which specify the addressed challenges in order to preserve the IoT data privacy from an end-to-end perspective.

4.3 Design goals

After analyzing the lifecycle of the IoT data, we aim at addressing the IoT data privacy issue in a three-layered privacy model: the *User Sphere* that is completely in the control of the data owner where the data are collected, the *Joint Sphere* where the data owner shares the control with the data consumer, and the *Recipient Sphere* that is completely out of the data owner's control and the data are processed by the data consumer. Thus, we identify below for each sphere a set of privacy requirements to be addressed in order to preserve the data owner's privacy.

Several researchers have adopted blockchain for non-monetary applications, such that enhancing preserving-privacy of IoT devices in the *User Sphere*. However, applying the blockchain technology to the IoT is not straightforward. Several challenges need to be addressed. First, the proof-of-work needs to be eliminated in order to decrease the transaction processing overhead. Indeed, this computationally expensive consensus is important for cryptocurrency to prevent double spending. This latter is not considered for IoT device management. Second, to enforce the data owner's control over the owned IoT devices, a behavior tracking is required in order to detect any possible misbehavior. Smart contract can be explored in this context to enforce the data owner's privacy preferences about how the IoT devices must behave.

In the *Joint Sphere*, IoT data control is shared between the data owner and the data consumer. In this context, the GDPR [GDPR, 2016] requires that the data consumers provide much stronger consent controls to the data owners. Due to the distributed nature of the IoT domain, security and privacy are considered as the major challenges of the IoT domain. The first challenge is to manage the IoT network participants' identities while preserving their privacy. To tackle this challenge, both anonymity and pseudonymity disguise the user's identity, thus need to be used to hide the connection between the real identity and the used pseudonyms in the IoT network. However, using both anonymity and pseudonymity is not enough to preserve privacy. Indeed, pseudonyms can be linked with the real identities by matching the individuals' profiles with their behaviors. Thus, using multiple pseudonyms by each IoT network participant is needed to enforce unlinkability. The three privacy properties, namely anonymity, pseudonymity, and unlinkability [Pfitzmann and Köhntopp, 2001] can be ensured thanks to the blockchain technology. Moreover, recent research proves the blockchain's potential to enforce privacy requirement compliance. Thus, a privacy-preserving solution needs to take advantage of the smart contract that can enforce a common agreement between several untrusted parties while eliminating the single point of trust issue. Indeed, the smart contract is needed to enforce the users' privacy choices and ensure that the shared data will be handled as expected in the whole IoT data lifecycle, namely the data collection, transmission, storage, and processing phases.

In the *Recipient Sphere*, only the data consumer who controls the data. In order to prevent

the consumer from learning individual data, several users can collaborate among them and aggregate their IoT data before sending the obtained result to the data consumer. Although multiple researchers have studied the IoT data aggregation field, many challenges remain to be addressed in order to tackle the privacy-preserving issue in this field. First, a distributed data storage is needed to eliminate the single point of failure problem that consists in storing, aggregating, and analyzing all the produced data by a centralized authority. Second, an end-to-end encryption data aggregation is needed to overcome both the single point of trust and the raw data disclosure issues that consist of giving all the raw data produced by smart devices to a trusted party to be aggregated. Finally, a privacy-preserving solution needs to take advantage of the asymmetric encryption, the hash functions, and the digital signature in order to guarantee the three security properties, namely the data confidentiality, the data integrity, and the sender's identity checking (i.e., authentication data).

To the best of our knowledge, none of the existing approaches considered all the privacy requirements mentioned above, while covering the whole IoT data lifecycle, from the user's consent to the data analysis. For this purpose, we propose PrivBlockchain, an end-to-end, distributed privacy-preserving framework for IoT data. Indeed, our contribution guarantees three security properties, namely the data confidentiality, data integrity, and sender's identity checking. Moreover, the three privacy properties, namely anonymity, pseudonymity, and unlinkability are ensured by our framework.

After considering the challenges of preserving IoT data privacy and defining our design goals, we detail in the next section our proposed IoT data privacy-preserving framework.

4.4 PrivBlockchain: IoT data privacy-preserving framework

Considering the legal rights imposed by the GDPR [GDPR, 2016], it is necessary to ensure the privacy requirement compliance to preserve privacy during the whole data lifecycle, covering the collection, transmission, storage and processing phases. Thus, we focus on (i) how to enforce the user's control over the owned smart devices, (ii) how to ensure the privacy requirement and obligation compliance between untrusted parties in an IoT environment, (iii) how to organize the smart devices into groups according to their owner's privacy choices, and (iv) how to keep the accuracy in data analytics using group-level aggregation. To this end, we propose PrivBlockchain, an end-to-end privacy-preserving framework for the IoT data based on semantic, blockchain, and homomorphic technologies. PrivBlockchain aims at addressing the aforementioned design goals in order to preserve the user's privacy in the IoT environment.

In this section, we first provide an overview of PrivBlockchain, then we define the core components of the PrivBlockchain framework. After that, we present the PrivBlockchain's modules.

4.4.1 PrivBlockchain overview

In order to manage the IoT devices, a data owner can introduce some privacy permission settings that define how each IoT device must behave according to the produced data based on our

defined LIoPY ontology (see Chapter 2). However, these permission settings need to be enforced to keep the data owner control over the owned devices. We aim at addressing this dilemma by proposing a framework that monitors the IoT devices by allowing or blocking access according to the devices' behaviors. Thus, our PrivBlockchain is based on: (i) a lightweight blockchain that eliminates the proof-of-work in order to be supported by the resource-constrained IoT devices, (ii) a smart contract that verifies the privacy permission settings before allowing any device to communicate with other devices, and (iii) a smart contract that analyzes the IoT devices' behaviors in order to detect any malicious attempt and rapidly block the detected devices.

From another side, the consumers need to collect and analyze the produced data from the smart devices to provide better facilities for the users. However, IoT data analytics increase the user's worries about the potential uses of collected data. In fact, the users desire to preserve their privacy while taking advantage of the offered services. Nevertheless, the users' IoT data can be shared with consumers for two purposes, namely first use purpose that consists in sharing individual-related data and second use purpose that consists in sharing anonymous data.

For the first use purpose (e.g., smart grid billing, patient's treatment, etc.), the users need to share their IoT data with authorized parties. However, the users have a little or no control over their IoT data once shared. Therefore, we aim at improving the data ownership, transparency, and auditability for users. Thus, our PrivBlockchain is based on the main following principles. **User-driven and transparency:** The user is the master of the own data thanks to the full control over the shared data in the network. **Fairness:** Using the blockchain in our end-to-end privacy-preserving framework improves fairness because nobody can systematically be enforced to lose control over the own data. **Distributed architecture and the lack of a central authority:** Each node in the network directly shares its data with other nodes, without the intervention of any third or trusted entity to manage the whole network. **Fine-granularity:** The use of a smart contract enables the user to implement expressive and granular privacy policies.

For the second use purpose (e.g., smart grid statistics, governmental health programs, and scientific research), several users can collaborate among them and aggregate their IoT data in order to prevent the consumer to learn individual data. However, aggregating IoT data requires a network manager and a trusted aggregator. We aim at addressing this dilemma by introducing a framework that improves the users' privacy while keeping the data accuracy. Thus, our PrivBlockchain framework is based on: (i) the blockchain technology that acts as a distributed data storage, (ii) the smart contract that acts as a data aggregation controller, and (iii) the homomorphic encryption technology that enables the computation over encrypted data, thus eliminates the need to trust a consumer or an aggregator.

Figure 4.1 depicts the proposed architecture of the PrivBlockchain framework. It is a smart space network that consists of three networks, namely private area, regional area, and blockchain. The private area network can be a smart home, a smart building, or a hospital. It includes several smart devices, namely data producer and semantic IoT gateway. This latter is a high resource device, which is responsible for the other owned data producers by each data owner that can be an individual or an organization. The data owner creates some smart contracts in order to

define the privacy permission settings of each data producer and control the producers' behaviors. The communication between the private area network's smart devices is stored in a private blockchain, called the private ledger. Moreover, the semantic IoT gateway belongs to both the blockchain and regional area networks.

The regional area network includes several semantic IoT gateways. These gateways can be organized into groups. Any semantic IoT gateway can be chosen to be the group's aggregator. This latter's role is to aggregate the data of the group members and then send the computed result to a smart contract hosted on a public blockchain that resides on the blockchain network.

The blockchain network includes several nodes, namely data consumer, data storage, and key generation node. The data storage node is a blockchain network node that provides a storing service for both public blockchain and data produced by the data producers. The consumer creates a smart contract in order to receive some aggregated data as a request's result. Each consumer, which can be an energy substation, a traffic routing station, or scientific researchers needs to associate a key generation node with the smart contract at deploying time. The key generation node is responsible for generating the keys. The communication between the different nodes in the blockchain network is stored in a public blockchain.

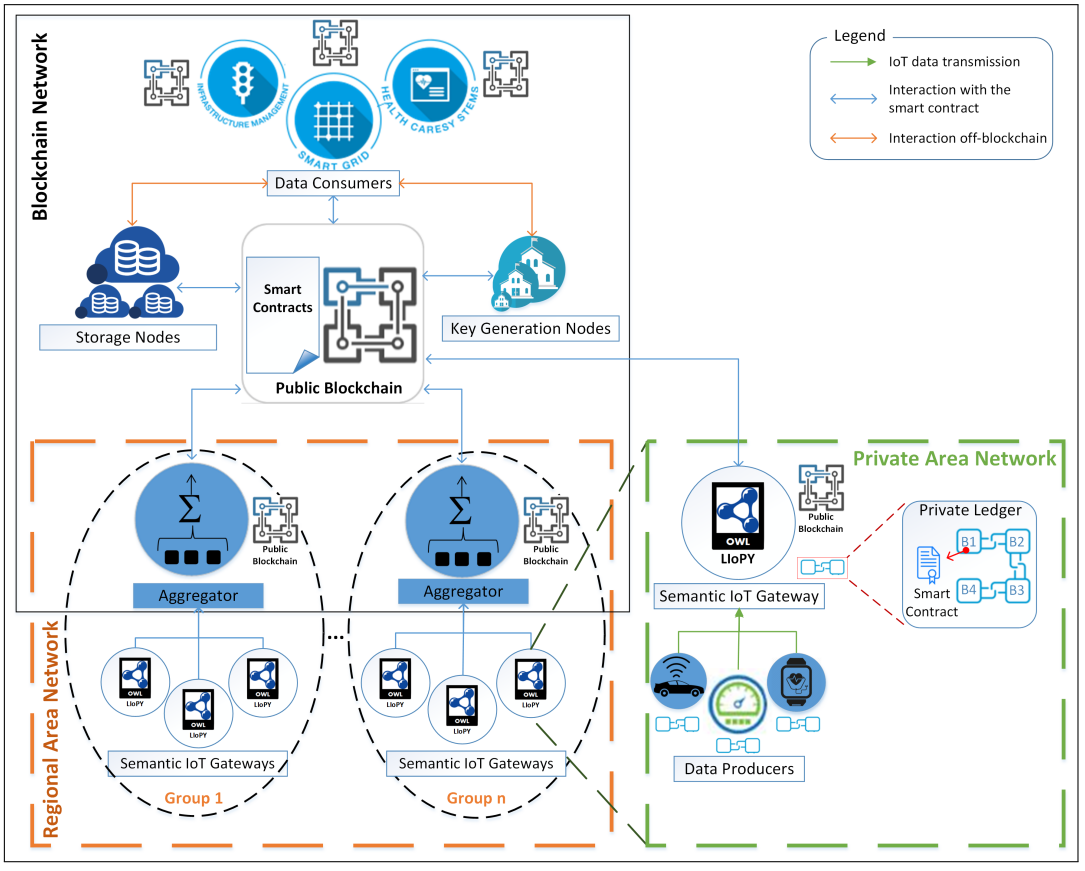


Figure 4.1: PrivBlockchain Architecture

After presenting an overview of the PrivBlockchain framework, we outline the proposed framework core components in the following section.

4.4.2 PrivBlockchain core components

As aforementioned, PrivBlockchain consists of several components, namely private area network, private ledger, regional area network, blockchain network, public blockchain, smart contract, data producer, group, key generation node, data consumer, and semantic IoT gateway.

The detailed description of these components is as follows:

4.4.2.1 Private area network

The private area network includes a set of smart devices, called data producers that are owned by a smart home's owner or a hospital's manager and controlled by one or more semantic IoT gateway nodes. These latter are high resource devices that validate communication between the data producers and link these IoT devices with both the regional area and the blockchain networks. In the private area network, we distinguish between two node types. The first type, called *full nodes* process every transaction and store the entire blockchain. The second type, called *light nodes* only store the relevant information, such as the addresses of the semantic IoT gateway and smart contracts due to their limited resources. Indeed, the semantic IoT gateways are *full nodes* while the data producers are *light nodes*. Moreover, each private area network maintains a private ledger.

4.4.2.2 Private Ledger

A private ledger is a local private blockchain that enables the data owner to control the own smart devices. This blockchain contains the data owner's data producers communication and has a set of smart contracts that enforce the data owner's privacy preferences on how the owned IoT devices must behave. In the private ledger, blocks are chained together using the hash of the previous block to keep the blockchain immutable. However, classic blockchains are computationally expensive and involve high bandwidth overhead and delays, which are not suitable for most IoT devices. To this end, we propose to use a lightweight blockchain that eliminates solving the proof-of-work used for mining new blocks into the classic blockchain. Therefore, a new block is created for each transaction and added to the private ledger, which reduces the block validation processing time while maintaining most of the classic blockchain security and privacy benefits. Moreover, the blockchain use leads to storage overhead cost. To overcome this issue, the private ledger stores only the newer blocks. Indeed, the semantic IoT gateway can only save the hash of the previous blocks and not the entire blocks in order to keep the blockchain immutable. The private ledger is kept and managed by a set of semantic IoT gateways that are also considered as nodes in the regional area network.

4.4.2.3 Regional area network

The regional area network can be a smart city, a smart grid, or any smart space that includes several smart devices that are able to bear the blockchain technology. In our case, we consider a regional area network as a set of semantic IoT gateways that are geographically close. These

gateways can be organized into groups in order to increase each member privacy. Indeed, they collaborate among them indirectly thanks to the consumer's smart contract that is hosted on the public blockchain maintained by a blockchain network.

4.4.2.4 Blockchain network

The proposed blockchain network is a peer-to-peer network (P2P) that contains several nodes with different memory and storage capabilities. The network's nodes can be a semantic IoT gateway, an aggregator, a data consumer, a storage node, or a key generation node. These nodes require high memory and storage capabilities to store the public blockchain. Each blockchain network node has several blockchain addresses and pair of public (PK) and private (SK) keys. The node's blockchain address, which is known by the other nodes in the blockchain network, is used as a unique node identifier to send transactions to the smart contracts or receive transactions from other nodes while the node's private key, which is kept secret to the node, is used to sign transactions before sending them. Then, the signature is verified using the node's PK in the transaction. The digital signature, which is the hash of a digital asset (i.e., a transaction) improves (i) the transaction sender's authentication (i.e., proves that the transaction sender has the appropriate couple of public key and blockchain address), (ii) the non-repudiation principle (i.e., the sender cannot deny having sent a transaction), and (iii) the transmitted message integrity (i.e., proves that a transaction is not altered while transmitted). Only valid transactions can be added to the public blockchain and distributed between all the network nodes. Moreover, the blockchain network maintains a public blockchain.

4.4.2.5 Public Blockchain

The public blockchain can be seen as the history of all the transactions that are sent by the nodes of the blockchain network to access or share IoT data. In fact, it can ensure auditing functions. Hence, our solution offers a non-repudiation principle compliance, which consists in preventing any blockchain network node from denying actions that are performed by itself. Furthermore, the public blockchain tackles both the first use purpose and the second use purpose by hosting two different type of smart contracts.

4.4.2.6 Smart contract

The smart contract can be seen as a published agreement within the blockchain that ensures the compliance of a set of conditions shared between untrusted parties. The smart contract contains a code and defines a set of functions. The public blockchain contains two smart contract types. The first smart contract, called IoTDataSharing enforces the data owner's privacy preferences on how the own data must be handled. In fact, the IoTDataSharing smart contract can be considered as a data owner's privacy policy that specifies some obligations for handling the shared IoT data. This contract is considered for the first use purpose whereas for the second use purpose we propose another smart contract, called IoTDataAggregation. This latter improves data accuracy in data analytics using group-level aggregation. It is created by one consumer in order to receive

aggregated data as a request's result from a group of smart devices. The IoTDataAggregation smart contract enables the consumer to define its terms of service and the users to introduce their privacy choices for their data producers.

4.4.2.7 Data producer

The data producer is an IoT device equipped with sensing and communication capabilities that allow it to collect environmental data, communicate with other devices, or connect to the Internet. In an IoT environment, we distinguish several devices, such that RFID readers, sensors, actuators, embedded computers, and mobile phones. Memory and storage capabilities differ from one device to another. In this work, the data producers are considered as smart devices with low memory and storage capabilities. Thus, they can delegate complicated treatments to the semantic IoT gateway in order to add transactions to the private ledger, generate a common privacy policy with the consumer, interact with the public blockchain's smart contracts, and store a copy of the public blockchain. In the regional area network and according to the produced IoT data, data producers and semantic IoT gateways can be organized into different groups.

4.4.2.8 Group

The group is a set of smart devices and an aggregator, which is a randomly chosen semantic IoT gateway. The aggregator's purpose is to obfuscate the individual IoT data of each of its group members by computing an aggregated result from the members' data. Groups are formed based on the members' privacy choices, which are stored on the consumer's smart contract. In order to eliminate the raw data disclosure issue, the homomorphic encryption technology is used. Thus, each group has a public key based on the Paillier cryptosystem [Paillier, 1999] and generated by the key generation node.

4.4.2.9 Key Generation Node

The key generation node is responsible for generating all the entity keys. Indeed, it generates several blockchain addresses, public and private keys for each smart device. Besides, it generates a public key Pk_{Pai} and a secret key Sk_{Pai} based on the Paillier cryptosystem [Paillier, 1999] in order to enable additive computation over encrypted data for each created group. Then, it updates the group's public key on the smart contract and shares the group's private key with the data consumer.

4.4.2.10 Data consumer

The data consumer can be a doctor or an emergency service crew that need IoT data for first use purpose or rather an energy substation, a traffic routing station, or a scientific researcher who needs IoT data for analytics purpose. In both cases, the consumer shares a smart contract with the data owners before handling the shared data. Each data consumer needs to share its terms of service that describe how its systems will handle the obtained data. The communication

between the data consumer and the data owner is established thanks to the owner's semantic IoT gateway.

4.4.2.11 Semantic IoT Gateway

Typically, gateways collect and send the collected data from IoT devices like sensors and actuators to external platforms in order to be remotely analyzed. In order to enable a better control over their data producers, a more flexible gateway is required by the users. For this purpose, we propose a Semantic IoT Gateway that aims at computing complicated treatments, such that logging the IoT devices communications in a private ledger in order to detect any misbehavior, generating a common privacy policy between the producer and the consumer, and relaying the private area network with both regional area and blockchain networks. Our Semantic IoT Gateway is based on LIoPY, the European Legal compliant ontology for supporting preserving IoT PrivacY defined in Chapter 2 and our reasoning process defined in Chapter 3.

The semantic IoT gateway is a smart device with high memory and storage capabilities. It belongs to the private area, regional area, and blockchain networks. In a private area network, the semantic IoT gateway's role is to be responsible for a set of data producers that have low memory and storage capabilities. Thus, it validates the incoming and the outgoing transactions before adding them to the private ledger. Moreover, it receives the produced data, generates a common privacy policy between the producer and the consumer, and acts as a relay between the data producers and the blockchain network. Indeed, the semantic IoT gateway is considered as a node in the blockchain network. It communicates with the blockchain network's nodes and stores a copy of the public blockchain to benefit from the IoT applications that are offered by the data consumers. For this purpose, it uses different couple of public and private keys in order to reduce the linkability problem. Moreover, the semantic IoT gateway is considered as a node in the regional area network. It connects the data producers to the data consumer. Thus, it participates to a group and sends the produced data by the producers to be stored on the consumer's smart contract. Moreover, the semantic IoT gateway has another role in the regional area network. It can be randomly chosen to be the group's aggregator. Thus, it aggregates all the members' data of a group, then records the result into the blockchain by updating the result value stored on the smart contract.

The architecture of the Semantic IoT Gateway is shown in Figure 4.2. It includes five core components, which are: (i) the Semantic Rule Manager, which aims at matching the data owner's privacy preferences and the data consumer's terms of service in order to generate an adapted privacy policy, (ii) the asymmetric encryption engine, which retrieves the group's public key from the smart contract, communicates this key to the data producers that encrypt their IoT data and send them back to the engine that retrieves the group's aggregator's public key, uses it to encrypt the received IoT data ciphertext, and publishes the encrypted data on the smart contract, (iii) the verification engine, which enables the semantic IoT gateway, when it is chosen to be a group's aggregator to retrieve all the group members' data and check the data integrity and the sender's identity of each member, (iv) the aggregation engine, which enables the semantic IoT

gateway to decrypt all the group's data using its secret key as an aggregator, aggregates the group's data without revealing their plaintexts, and publishes the encrypted result on the smart contract, and (v) the Blockchain Client, which is considered as a miner of the private ledger and an access point to the blockchain network to receive a blockchain address and access the public blockchain. All the Semantic IoT Gateway components interact among them and with the private area and blockchain networks in order to preserve the IoT data privacy.

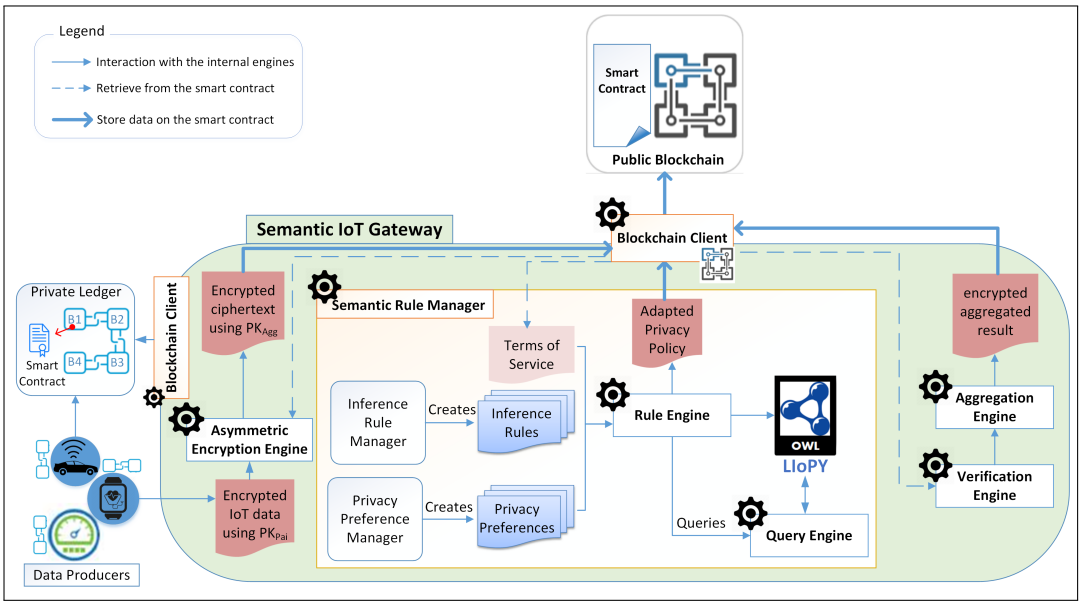


Figure 4.2: Architecture of Semantic IoT Gateway

After defining the system model's core components, we introduce below the three modules of the PrivBlockchain framework.

4.4.3 PrivBlockchain modules

PrivBlockchain includes three modules, namely blockchain-based IoT device management, blockchain-based IoT data sharing, and homomorphic encryption-based IoT data aggregation.

The first module enables the data owner to control the IoT devices by defining the privacy permission settings about how each device must behave, logging the communication between the devices in a private area on the private ledger, and checking the IoT device's behavior before allowing it to communicate with other devices or connect to the Internet.

After addressing the data owner's control over the smart devices' behaviors in the private area, we tackle the data owner's control over the shared data with the data consumers. In the IoT domain, data can be shared between multiple involved parties for two purposes. The first is sharing data for a first use purpose, such as smart grid billing, patient's treatment, etc. Whereas the other purpose is sharing data for a second use purpose, such as smart grid statistics, governmental health programs, and scientific research.

For the first use purpose, the second PrivBlockchain's module aims at enforcing a common agreement between untrusted parties by converting the user's privacy choices to a smart contract

that can enforce the user's control over the shared data by monitoring how the data are handled and having the ability to add or revoke authorization to the consumers.

Although the second module improves the data ownership and transparency for users, it cannot eliminate the raw data disclosure issue when dealing with data processing for the second use purpose. In this context, we propose the third module that is based on both blockchain and homomorphic encryption technologies in order to preserve the IoT data during the whole lifecycle. Thus, this module enables IoT data aggregation at group-level.

In the next sections, we detail the three PrivBlockchain framework's modules, namely blockchain-based IoT device management (Section 4.5), blockchain-based IoT data sharing (Section 4.6), and homomorphic encryption-based IoT data aggregation (Section 4.7).

4.5 Blockchain-based IoT device management module

According to [Clarke, 2006], the privacy of the person is the right to control the integrity of the body and the wearable IoT devices. In order to guarantee this right, we propose to use blockchain and semantic to define a blockchain-based IoT device management module. This module aims at enforcing the data owner control over the IoT devices by detecting any possible misbehavior.

In this section, we describe our proposed smart contracts and detail the privacy permission setting adding process.

4.5.1 Smart contract description

In order to enforce the privacy permission settings, three smart contracts are proposed, namely PrivacyPermissionSetting, Ownership, and BehaviorControl. These contracts enforce the data owner's privacy preferences on how the IoT devices must behave according to each data output.

PrivacyPermissionSetting smart contract: it is created by the semantic IoT gateway and hosted on the private ledger. Each IoT device that knows the smart contract address can use it by invoking its defined functions. The PrivacyPermissionSetting smart contract is designed to store the permission for each IoT device concerning a specific IoT data output according to the data owner's privacy preferences. This smart contract defines a set of functions, namely: (i) **LocalStore** function that enables to verify the IoT device permission to locally store its collected data, (ii) **ExternalStore** function that verifies if the IoT device has the permission to send the collected data to be stored on an external storage node, (iii) **Read** function that verifies if the IoT device has the permission to request data from other internal or external IoT devices after verifying the IoT device permissions, (iv) **Write** function that enables an IoT device to add and/or modify a requested data collected by other internal or external IoT devices if the IoT device is permitted, and (v) **Monitor** function that enables to verify the IoT device permission to receive periodic data from another IoT device. Furthermore, the PrivacyPermissionSetting smart contract includes a self-destruct function. Only the data owner's address can invoke this function to destruct the smart contract in order to revoke the granted privacy permissions for all the IoT de-

vices associated with this contract. It is worth noting that when destructing the smart contract, it will be inoperable, but its history remains in the private ledger.

Ownership smart contract: it is created by the semantic IoT gateway in order to store its own IoT devices' addresses. For each IoT device, a set of IoT data outputs is added. A PrivacyPermissionSetting smart contract is associated with each IoT device output. Thus, the smart contract address is stored on the Ownership smart contract and sent to the appropriate IoT device according to its data outputs and granted permissions. The Ownership smart contract is designed to enforce the data owner's control over the IoT devices and their outputs. It defines a set of functions, namely: (i) **addNewIoTDevice** function, which enables to add a new IoT device by indicating an IoT device address, an IoT device output, and the address of the PrivacyPermissionSetting smart contract, which is associated with the IoT device output, (ii) **modifyIoTDevice** function, which enables to modify the description of an existing IoT device except for the set of its outputs, (iii) **removeIoTDevice** function, which enables to remove an existing IoT device, (iv) **addIoTDeviceOutput** function, which enables to add a new output to an existing IoT device by indicating a description output and the associated PrivacyPermissionSetting smart contract address, (v) **modifyIoTDeviceOutput** function, which enables to modify the description of an existing IoT device output, and (vi) **removeIoTDeviceOutput** function, which enables to remove an existing IoT device output from an existing IoT device.

BehaviorControl smart contract: it is created by the semantic IoT gateway in order to define the privacy settings for each IoT device output, verify the permissions before allowing any device to communicate with other devices or connect to the Internet, and block an IoT device access to a resource in case of sending too many requests during a very short time. The BehaviorControl smart contract is designed to rapidly detect any malicious attempt by analyzing the IoT device behavior. It defines a set of functions, namely: (i) **privacySettingAdd** function, which enables to add a new privacy setting for an IoT device according to its data output by introducing the action to be handled on the data, its permission, and its allowed frequency threshold, (ii) **privacySettingUpdate** function, which enables to modify the permission associated to the action on the output of one IoT device, (iii) **privacySettingDelete** function, which enables to revoke the permission from an existing IoT device, (iv) **misbehaviorPenalty** function, which enables to compute the duration time of an IoT device's penalty when a misbehavior is detected, and (v) **verifyPermission** function, which enables to check the IoT device's behavior before allowing the transaction's sender to access to the requested data output.

Based on the defined smart contracts, we introduce below the privacy permission setting adding process.

4.5.2 Privacy permission setting adding process

Figure 4.3 presents the process of adding a new IoT device by a data owner to the private area network and defining some privacy preferences about how this new IoT device must behave. This process enables the semantic IoT gateway node to add a new IoT device with its PrivacyPermissionSetting smart contract to the private ledger.

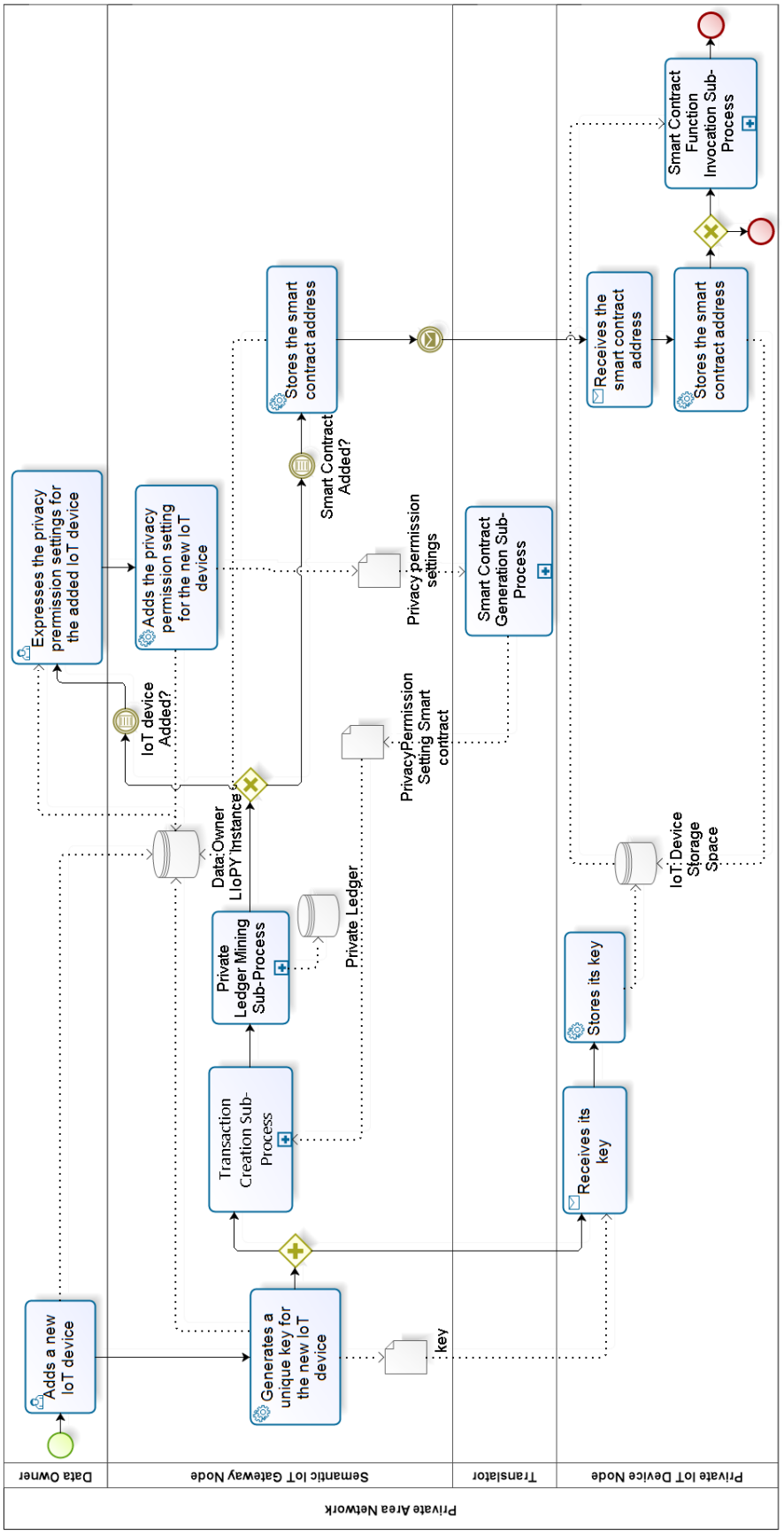


Figure 4.3: Add new IoT device business process notated in BPMN

The process begins when the data owner adds a new IoT device to the "Data Owner LIoPY Instance" base. Then, the semantic IoT gateway node generates a unique key for this new IoT device and sends the key to this device, which stores its key on its local storage space. Besides, the semantic IoT gateway node stores this key on the "Data Owner LIoPY Instance" base and executes the transaction creation sub-process (see Figure 4.4), which creates a signed transaction that contains the IoT device key and sends it to the private ledger mining sub-process (see Figure 4.5). This latter updates the private ledger by adding a new transaction once it is checked. If the IoT device is added, the data owner expresses some privacy preferences for the added IoT device. Then, the semantic IoT gateway node adds the privacy permission settings to the "Data Owner LIoPY Instance" base, sends it to the translator that executes the smart contract generation sub-process (see Figure 4.6), and receives the appropriate smart contract. After that, the semantic IoT gateway node executes the transaction creation and the private ledger mining sub-processes. If the smart contract is added to the private ledger then the semantic IoT gateway node stores the smart contract address on the "Data Owner LIoPY Instance" base and sends it to the IoT device that stores the address on its local storage space. Finally, the private IoT device node can stop the process or it can execute the smart contract function invocation sub-process (see Figure 4.7).

We detail below the four used sub-processes, namely transaction creation, private ledger mining, smart contract generation, and smart contract function invocation.

Figure 4.4 shows the transaction creation sub-process in details. When the semantic IoT gateway node receives the transaction inputs, it creates the transaction. Then, it uses its private key in order to sign the created transaction. Once signed, the transaction is sent to the private area network nodes to be checked/added to the private ledger.

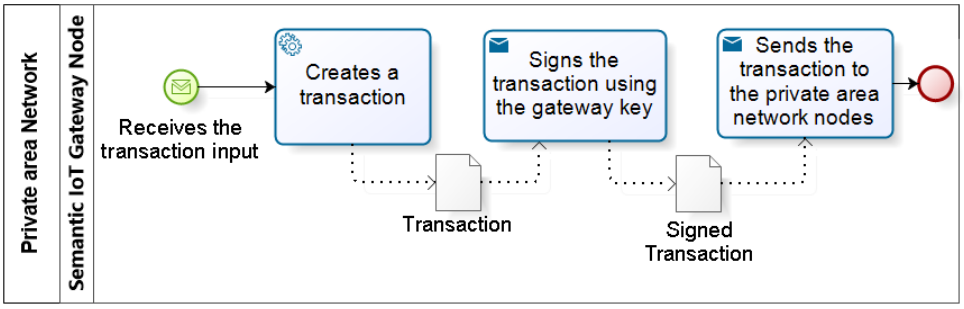


Figure 4.4: Transaction creation sub-process notated in BPMN

Figure 4.5 shows the private ledger mining sub-process in details. The sub-process begins when the semantic IoT gateway node receives a signed transaction. Then, it checks the transaction validity. In case of a valid transaction, the semantic IoT gateway node creates a new block that contains the transaction and the hash of the previous block, adds the block to the private ledger, then the sub-process is successfully finished. If the transaction is invalid, it is rejected and the sub-process is stopped.

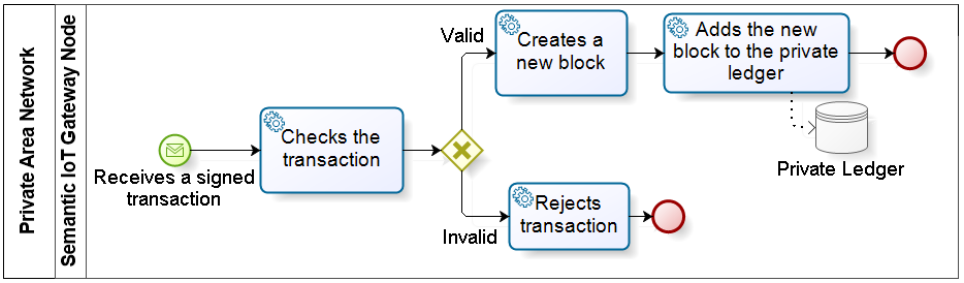


Figure 4.5: Private ledger mining sub-process notated in BPMN

Figure 4.6 shows the smart contract generation sub-process in details. In order to generate a new smart contract, the semantic IoT gateway node begins by retrieving the appropriate privacy permission settings from the "Data Owner LIoPY Instance" base. Then, it sends them to the translator, which will transform the privacy permission settings to a smart contract using the set of predefined functions. In fact, each privacy permission setting is presented by a function on the smart contract. Moreover, some functions are automatically added to any smart contract regardless of the privacy permission settings. For instance, the constructor function, which enables creating the smart contract itself is an example of these default functions. Once the smart contract is created, the translator sends it to the semantic IoT gateway node that stores it on the "Data Owner LIoPY Instance" base for future use and the sub-process is successfully finished.

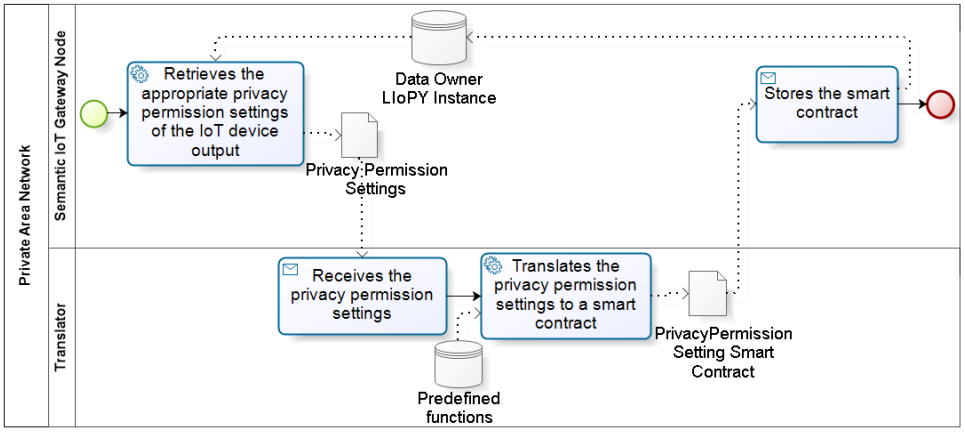


Figure 4.6: Smart contract generation sub-process notated in BPMN

Figure 4.7 shows the smart contract function invocation sub-process in details. This sub-process begins when an IoT device needs to invoke a function of its PrivacyPermissionSetting smart contract in order to handle an action on its collected data or communicate with the other IoT devices. First, the IoT device retrieves the smart contract address from its local storage space. Then, it creates a transaction that invokes the needed contract function and signs it using its key. After that, the IoT device sends its signed transaction to the semantic IoT gateway that executes the private ledger mining sub-process. In case of a valid transaction, the invoked smart contract function results will be stored on the private ledger and the sub-process is successfully finished.

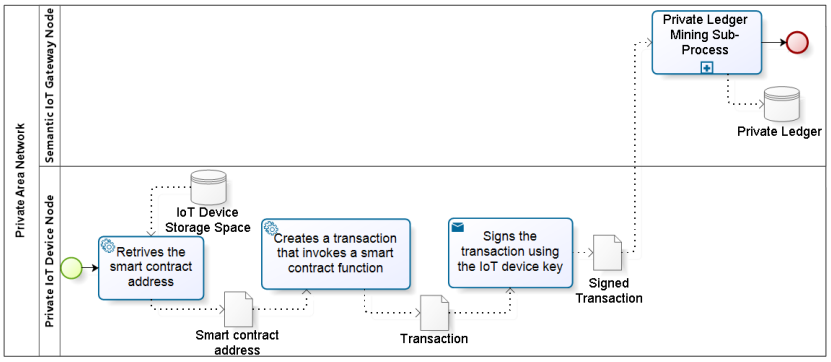


Figure 4.7: Smart contract function invocation sub-process notated in BPMN

As aforementioned before, each IoT device's behavior is tracking in order to detect any possible misbehavior. For this purpose, we propose two algorithms, namely misbehaviorPenalty and verifyPermission that are detailed below on Algorithm 2 and Algorithm 3, respectively.

Algorithm 2 aims at pushing the received misbehavior into a dynamic array that stores the detected misbehavior records of each IoT device and computing its duration time penalty when a misbehavior is detected. The misbehaviorPenalty Algorithm takes as input the subject (i.e., IoT device's blockchain address), the requested IoT device output, the asked action, the misbehavior type, and the time when the misbehavior is occurred. For each subject, a record array of misbehavior is stored and used to compute the penalty, which is the block duration opposed to the subject in terms of number of minutes. During these minutes, the subject cannot invoke any operation using its smart contract. The penalty is computed according the subject's record array of misbehavior and the subject's frequency threshold of invoking a specific action on one device output. Finally, the misbehaviorPenalty Algorithm returns the computed penalty.

Algorithm 2: IoT device misbehavior judge.

Input: *subject*, *deviceOutput*, *action*, *misbehavior*, *time*
Output: *penalty*
1 **Function misbehaviorPenalty** (*subject*, *deviceOutput*, *action*, *misbehavior*, *time*) **:**
2 *length* = MisbehaviorList[*subject*].length + 1
3 *penalty* = *length* / privacySettings[*subject*][*deviceOutput*][*action*].frequencyThreshold
4 MisbehaviorList[*subject*].push(Misbehavior(*subject*, *deviceOutput*, *action*, *misbehavior*, *time*, *penalty*))
5 **return** *penalty*
6 **End Function**

Algorithm 2 is used by Algorithm 3 in case of a misbehavior detection. Indeed, Algorithm 3 aims at checking the IoT device's behavior before allowing it to handle the requested action on the device output. Thus, it is executed each time an IoT device invokes a function of its PrivacyPermissionSetting smart contract. The verifyPermission Algorithm takes as input the subject, the device output, the asked action, and the time when the smart contract's function is invoked. It returns the request result and authorization message. First, a defined privacy permission setting that introduces the action for the couple of subject and device output needs to exist. Otherwise,

a misbehavior is detected, stored on the subject's record array of misbehavior, and the subject request is denied. If a privacy permission setting exists, both the subject and the output are verified if they are blocked. In case of unblock, both the IoT device's privacy permission setting and the IoT device's behavior are checked. If the permission is allowed and no misbehavior is detected, the verifyPermission Algorithm authorizes the action. Otherwise, a penalty is computed, the subject is blocked, and the permission is denied. Several misbehavior types can be detected by the verifyPermission Algorithm, such that sending requests to invoke unauthorized action on device output, sending requests during the penalty duration time, and sending multiple requests in a short period of time. The device output can also be momentary blocked to be protected from a possible attack when receiving multiple requests from multiple subjects in a short period of time.

Algorithm 3: IoT device misbehavior detection.

Input: *subject*, *deviceOutput*, *action*, *time* ▷ `subject invokes an action on one device output at specific time`
Output: *requestResult*, *authorizationMessage*

Function verifyPermission(*subject*, *deviceOutput*, *action*, *time*):
privacySettingCheck = false
behaviorcheck = true
penalty = 0
privacySetting= privacySettings[*subject*][*deviceOutput*][*action*]
outputSetting= outputSetting[*deviceOutput*]
if (! privacySetting.exists) **then**
  behaviorcheck = false
  penalty = **misbehaviorPenalty**(*subject*, *deviceOutput*, *action*," Unauthorized action attempt ! ", *time*)
  authorizationMessage= " Wrong subject specified ! "
else
  if (TimeofDeviceOutputUnblock[*deviceOutput*] $\geq$ *time*) **then**
    authorizationMessage= " Device Output are still blocked ! "
  end if
  if (behaviors[*subject*].TimeofSubjectUnblock $\geq$ *time*) **then**
    behaviorcheck = false
    penalty = **misbehaviorPenalty**(*subject*, *deviceOutput*, *action*," Successive failure ! ", *time*)
    authorizationMessage= " Subject is still blocked ! "
    ▷ `When the subject and the device Output are not blocked`
  else
    if (TimeofDeviceOutputUnblock[*deviceOutput*] $>$ 0) **then**
      TimeofDeviceOutputUnblock[*deviceOutput*] = 0
      outputSetting.frequentRequestsNumber = 0
      outputSetting.lastRequest = 0
    end if
    if (behaviors[*subject*].TimeofSubjectUnblock $>$ 0) **then**
      behaviors[*subject*].TimeofSubjectUnblock = 0
      privacySetting.frequentRequestsNumber = 0
      privacySetting.lastRequest = 0
    end if
    ▷ `Privacy permission setting check`
    if (privacySetting.permission == "*allow*")) **then**
      privacySettingCheck = true
    end if
    ▷ `IoT device's behavior check`
    if *time* - privacySetting.lastRequest $\leq$ privacySetting.minInterval **then**
      privacySetting.frequentRequestsNumber++

```
35:         if (privacySetting.frequentRequestsNumber ≥ privacySetting.frequencyThreshold) then
36:           behaviorcheck = false
37:           penalty = misbehaviorPenalty(subject, deviceOutput, action,"Too frequent request!", time)
38:           behaviors[subject].TimeofSubjectUnblock = time + penalty
39:           authorizationMessage= " Subject is blocked ! "
40:         end if
41:       else
42:         privacySetting.frequentRequestsNumber = 0
43:       end if
44:       privacySetting.lastRequest = time
45:       privacySetting.requestResult = privacySettingCheck and behaviorcheck
46:       if (time - outputSetting.lastRequest ≤ outputSetting.minInterval) then
47:         outputSetting.frequentRequestsNumber++
48:         if (outputSetting.frequentRequestsNumber ≥ outputSetting.frequencyThreshold) then
49:           TimeofDeviceOutputUnblock[deviceOutput] = time + outputSetting.frequencyThreshold
50:           authorizationMessage= " Data output are blocked ! "
51:         end if
52:       else
53:         outputSetting.frequentRequestsNumber = 0
54:       end if
55:       outputSetting.lastRequest = time
56:       if (privacySettingCheck and behaviorcheck) then
57:         authorizationMessage= " Action authorized ! "
58:       else if (!privacySettingCheck and behaviorcheck) then
59:         authorizationMessage= " Permission Denied ! "
60:       end if
61:     end if
62:  end if
63:  return ( (privacySettingCheck and behaviorcheck), authorizationMessage )
64:   End Function
```

After addressing the data owner's control over the smart devices' behaviors in the private area, we tackle the data owner's control over the shared data with the data consumers in the blockchain network in the following section.

4.6 Blockchain-based IoT data sharing module

Despite the increasing legislation pressure, several privacy requirements, such as the user's consent, purpose specification, and collection limitation, have been less addressed in the IoT domain. Thus, a privacy-preserving solution needs to take advantage of the smart contract that can enforce a common agreement between several untrusted parties without the involvement of a trusted third-party. For this purpose, we propose a blockchain-based module in order to prevent any privacy violation attempts by enforcing the users' privacy choices and ensuring that the shared data will be handled as expected in the whole IoT data lifecycle, namely the data collection, transmission, storage, and processing phases.

In this section, we describe our proposed smart contract and detail the IoT data sharing process.

4.6.1 Smart contract description

As aforementioned, we propose for this module a smart contract, called IoTDataSharing whose owner is the user. This smart contract acts as an agreement between the owner and the consumer. It enforces the data owner's privacy preferences and requirements on how the own data must be handled once shared.

IoTDataSharing smart contract: it is created when a data owner wants to share new IoT data with consumers. A set of subscribers can be added to the allowed consumer's list. This smart contract is designed to enforce the data owner's privacy preferences on how the data must be handled once shared. The IoTDataSharing smart contract contains many data fields, such as data hash, creation date, and a set of consumer's blockchain addresses. Each consumer receives the hash of the pointer of the file location in the off-blockchain storage through a transaction sent by the data owner. Moreover, each consumer is defined with various permissions. The defined **addConsumer** function enables to add a new consumer by indicating its blockchain address, and a set of permissions relevant to the privacy requirements, such as the allowed action according to the chosen purpose, operation, retention duration, disclosure limitation, etc. Moreover, a set of conditions can be added to each existing consumer's permission, such as the allowed location consumer's address, time of day for handling the shared data, and the allowed role of the consumer's address. To this end, an **addCondition** function is defined. When the retention duration ends, the consumer's address is automatically removed from the consumer's list by invoking the **removeConsumer** function, thus it is no longer notified by new shared data. Besides, this function is used when the data owner wants to revoke the permissions of a specific consumer. In case of the file content modification, the **updateFile** function needs to be invoked in order to keep consistency between the file hash that is stored on the blockchain and the off-blockchain stored file content. Similar to the smart contract owner, a consumer with *write permission* can invoke this function in order to change the hash of the file content. It should be noted that the use of the data hash enables the data integrity.

Based on the proposed smart contract, we detail below the dynamic aspect of the blockchain-based IoT data sharing module by introducing the process of sharing IoT data.

4.6.2 IoT data sharing process

Figure 4.8 presents the process of IoT data sharing with a data consumer according to its terms of service. The acceptance of the consumer's terms of service leads to create a data access authorization for the consumer and store it on the IoTDataSharing smart contract that is hosted on the public blockchain and used by the data consumer to handle the requested data.

The process begins when the data consumer node creates an instance of its terms of service, which contains the requested data, why, when, where, how, to whom, and for how long the data are needed. Then, the consumer creates a $T_{GetPermission}$ transaction while introducing the semantic IoT gateway node's blockchain address, adds the created terms of service instance to the transaction data, and broadcasts the transaction to the blockchain network's nodes. Only mining nodes in the blockchain network participate to solve the blockchain proof-of-work. Indeed,

they receive the transaction and start the blockchain mining sub-process (see Figure 4.9) in order to validate the received transaction. In case of a valid transaction, the mining nodes add this transaction to a new block to be stored on the public blockchain. When the semantic IoT gateway receives the $T_{GetPermission}$ transaction, it retrieves the consumer's terms of service instance and starts the terms of service treatment sub-process (see Figure 4.10). A successful treatment generates some privacy permissions that are a set of access authorization for the consumer to handle the shared data. Then, the semantic IoT gateway verifies if there is an already published file with the same requested data to retrieve the associated file details. Otherwise, it starts the file creation sub-process (see Figure 4.11) that returns the pointer hash of the file location. Then, the gateway invokes the GrantPermission transaction creation sub-process (see Figure 4.12) that takes as inputs both the consumer's privacy permissions and the file details. In case of successful transaction creation, this sub-process returns a signed $T_{GrantPermission}$ transaction that is broadcasted to the blockchain network nodes. The mining nodes receive the transaction and start the blockchain mining sub-process in order to validate the received transaction. Once the $T_{GrantPermission}$ transaction is mined, the semantic IoT gateway invokes the SendHash transaction creation sub-process (see Figure 4.13) that takes as input the pointer hash of the file location and returns a signed $T_{SendHash}$ transaction that is broadcasted to the blockchain network nodes. The mining nodes receive the transaction and start the blockchain mining sub-process in order to validate the received transaction. Once the $T_{SendHash}$ transaction is mined, the consumer retrieves the encrypted pointer hash of the file location from the $T_{SendHash}$ transaction's data and uses its private key to decrypt the pointer hash.

When the consumer obtains the pointer hash of the file location, it can invoke a fetch function to retrieve the data corresponding to the received pointer hash of the file location or send a $T_{GetSharedResource}$ transaction that invokes a set of the IoTDataSharing smart contract functions to handle the data. Before executing each function, the set of the consumer's permissions is verified to enforce the data owner's privacy preferences. For instance, if the consumer has the permission to disclose the file content and the retention duration is not finished yet, it can invoke the addConsumer function in order to add a new consumer to the file but with read-only permission and limited retention duration.

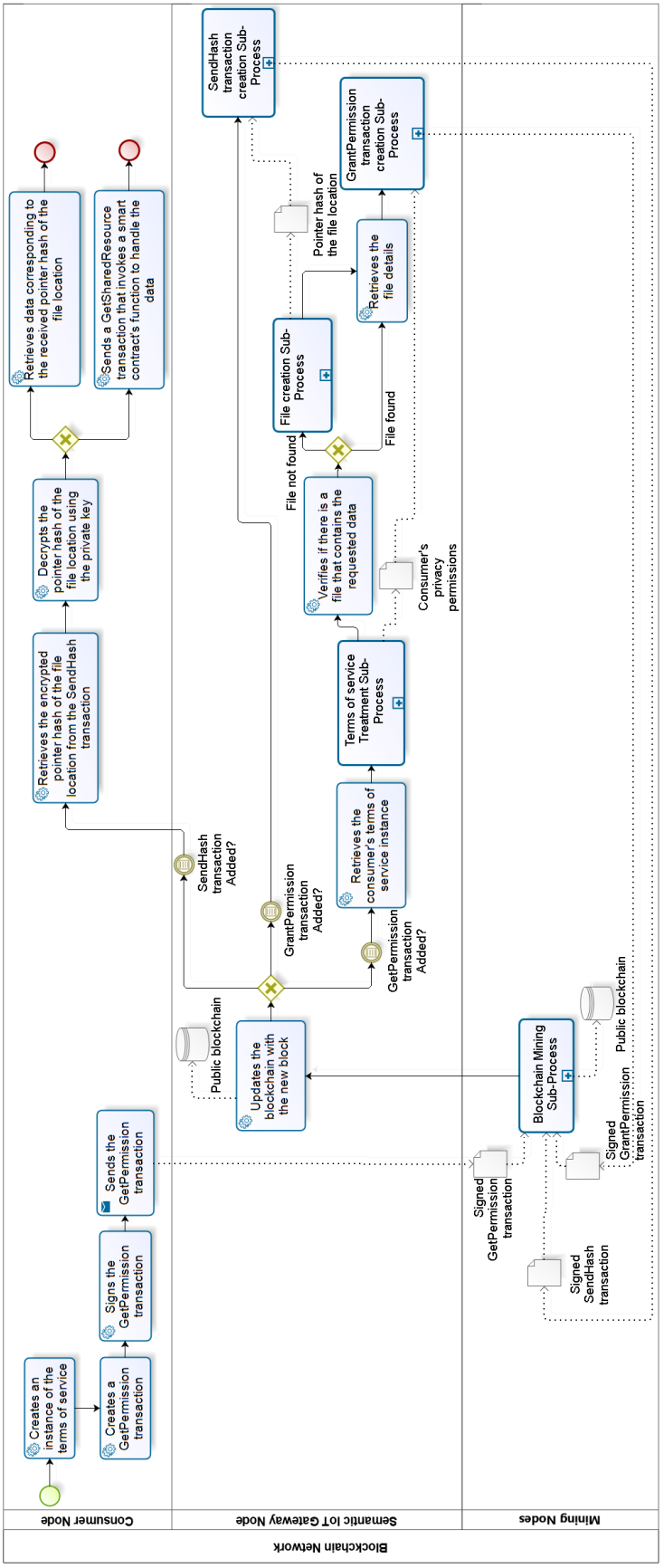


Figure 4.8: IoT data sharing business process notated in BPMN

We detail below the five used sub-processes, namely blockchain mining, terms of service treatment, file creation, grantPermission transaction creation, and sendHash transaction creation.

Figure 4.9 shows the blockchain mining sub-process in details. The sub-process begins when the mining nodes receive a signed transaction, they check the transaction validity in order to add it to the blockchain. In case of a valid transaction, the mining nodes mine a new block that contains a set of valid transactions and the hash of the previous block, add it to the public blockchain, and send it to the different blockchain network nodes to maintain the same copy of the public blockchain and the sub-process is finished with success. If the transaction is invalid, it is rejected and the sub-process is stopped.

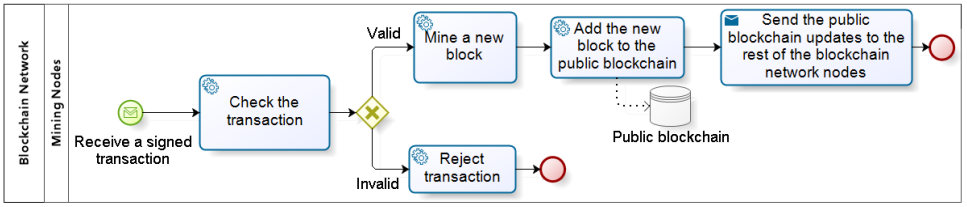


Figure 4.9: Blockchain mining sub-process notated in BPMN

Figure 4.10 shows the terms of service treatment sub-process in details. When the semantic IoT gateway node receives an instance of the consumer's terms of service, it communicates with the semantic rule engine, which is responsible for reasoning about the received request and then tacking a decision to create or not a privacy policy. First, the semantic rule engine retrieves the predefined set of inference rules, which are stored on the storage node and shared between all the involved parties. These inference rules help the engine to retrieve the appropriate data owner privacy rules from the "Data Owner LIoPY Instance" base. Then, the semantic rule engine matches the consumer's terms of service instance with the privacy rules in order to infer a privacy policy. In case of a match, a privacy policy is created by the semantic rule engine and sent to the semantic IoT gateway node that stores the privacy policy on the "Data Owner LIoPY Instance" base. Then, the semantic IoT gateway creates the consumer's privacy permissions and the sub-process is finished with success. Otherwise, the semantic rule engine rejects the consumer's terms of service instance and the sub-process is stopped.

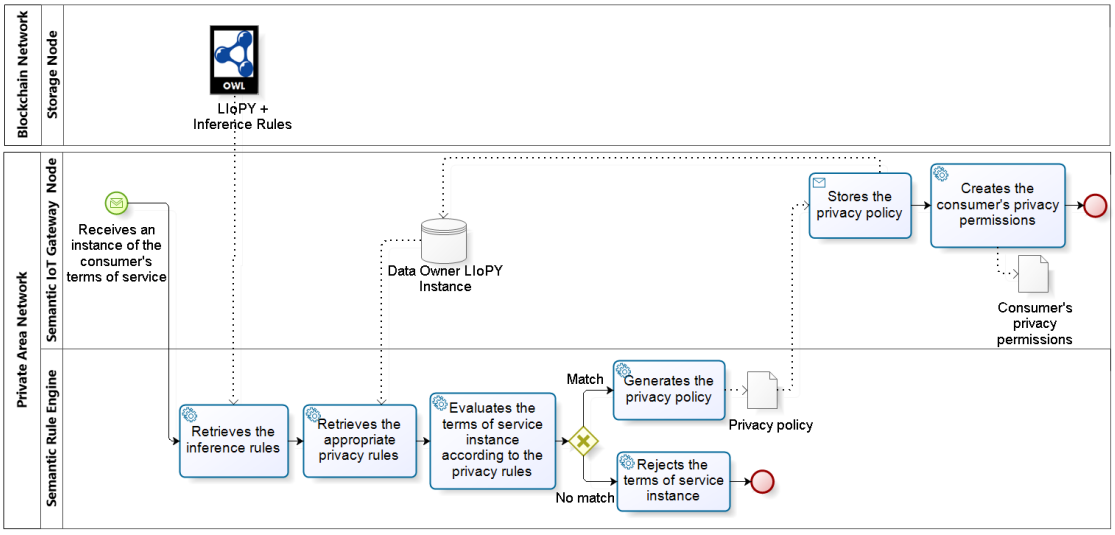


Figure 4.10: Terms of service treatment sub-process notated in BPMN

Figure 4.11 shows the file creation sub-process in details. The semantic IoT gateway starts by creating a new file that contains the result of the requested data. Then, it sends the file to a storage node to be stored. After that, the storage node generates the pointer hash of the file location and sends it back to the semantic IoT gateway, then the sub-process is finished with success. It is worth noting that an off-blockchain database is used to store the data, in our case we have used the InterPlanetary File System (IPFS) [Benet, 2014], as a peer-to-peer storage system used to overcome the expensive cost of storing data on the blockchain. Thus, only the hash pointer of the data location are exchanged by the blockchain's transactions.

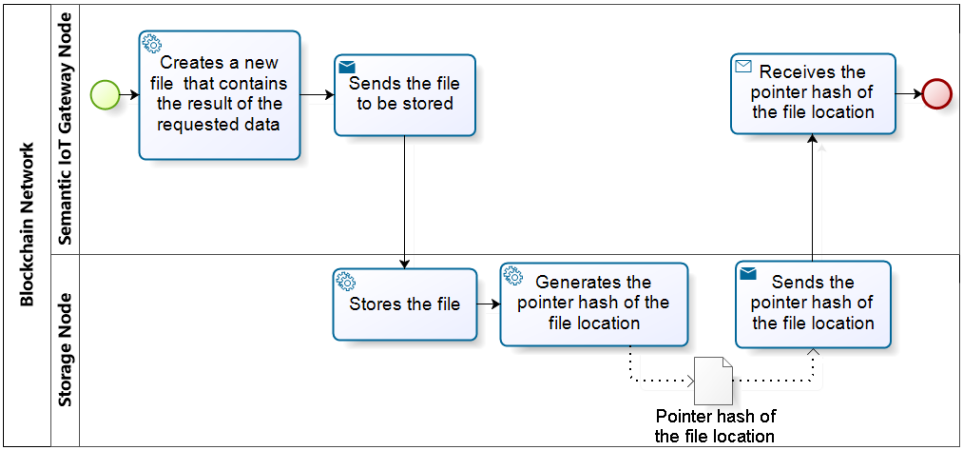


Figure 4.11: File creation sub-process notated in BPMN

Figure 4.12 shows the grantPermission transaction creation sub-process in details. Once the semantic IoT gateway receives the appropriate file data details and the consumer's privacy permissions, it creates a $T_{GrantPermission}$ transaction that invokes the addConsumer function of the IoTDataSharing smart contract with the appropriate privacy permissions. The $T_{GrantPermission}$ transaction enables to add a new consumer to the list of the allowed consumers of the shared

file. After that, the semantic IoT gateway signs the transaction and propagates it to the blockchain network and the sub-process is successfully finished.

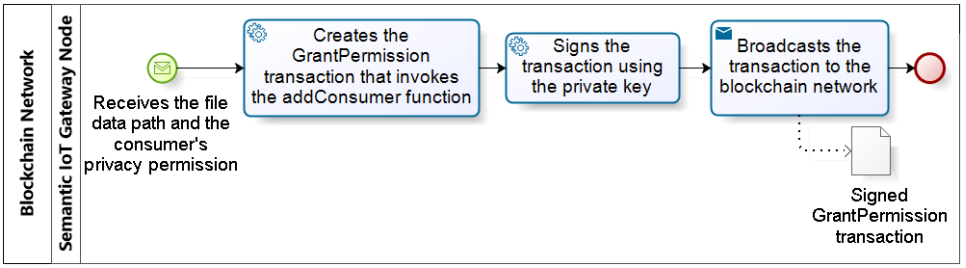


Figure 4.12: GrantPermission transaction creation sub-process notated in BPMN

Figure 4.13 shows the sendHash transaction creation sub-process in details. Once the semantic IoT gateway receives the appropriate pointer hash of the file location, it encrypts it using the data consumer's public key. Then, it creates a $T_{SendHash}$ transaction, adds the encrypted pointer hash to the transaction's data, signs the transaction, and propagates it to the blockchain network and the sub-process is successfully finished.

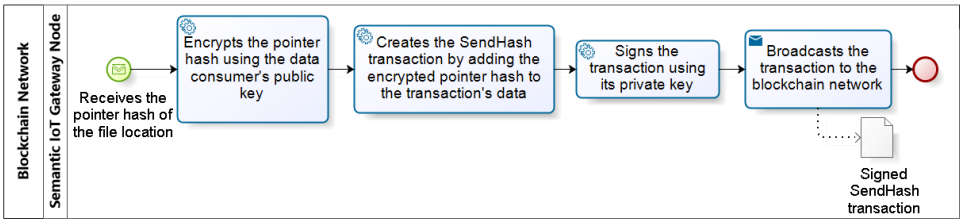


Figure 4.13: SendHash transaction creation sub-process notated in BPMN

After presenting the PrivBlockchain's module that addresses the first use purpose, we detail in the next section the last module that tackles the privacy issue for the second use purpose of IoT data.

4.7 Homomorphic encryption-based IoT data aggregation module

Although the blockchain-based IoT data sharing module improves the data ownership, transparency, and auditability for users, it cannot eliminate the raw data disclosure issue for the second use purpose of IoT data. In order to tackle this issue, we propose a smart contract that aims at organizing several users into groups based on their privacy preferences to collaborate among them and aggregate IoT data. Indeed, group-level aggregation prevents the consumer to learn individual data. However, aggregating IoT data requires a trusted aggregator. For this purpose, we propose to use the homomorphic encryption technology that enables any aggregator to obtain computation result over encrypted data calculation without knowing the appropriate plaintexts.

In this section, we describe our proposed smart contract and detail both the privacy policy generation and the IoT data aggregation processes.

4.7.1 Smart contract description

We propose a smart contract, called IoTDataAggregation whose owner is the consumer. This smart contract improves (i) common agreement enforcement between several untrusted parties, (ii) group-level aggregation of IoT data according to the users' privacy choices, and (iii) prevention of any identity fraud attempts concerning all the involved parties.

IoTDataAggregation Smart Contract: it is created when a consumer wants to receive an aggregated data from a set of IoT data producers. A set of groups can be created according to the privacy choices of the producers' owners. This smart contract is designed to enforce the data owner's privacy thanks to the group-level data aggregation. The IoTDataAggregation smart contract includes many data fields, namely terms of service, privacy policy, group, request, and producer. Each producer is associated with a privacy policy that defines various permissions. The defined **updateToS** function enables to add new terms of service by indicating the requested data and why, when, where, how, to whom, and for how long the data are needed. Moreover, a set of privacy requirements that matches the consumer's terms of service can be associated with each producer. To this end, an **updatePrivacyPolicy** function is defined. Each time the consumer wants to send a new request, it invokes both **CreateGroup** and **addParticipants** functions. This latter consists in deciding whether or not one data producer can be included into the group according to its associated privacy policy. Moreover, only the key generation node can invoke the **updatePK** function in order to share the public key generated based on the Paillier cryptosystem with the group members. These latters can retrieve and use the public key before sending their encrypted IoT data by invoking the **addParticipation** function. This function takes as input the group id, the participation, the participation's hash, and the participation's signature. The **verifyHashVal** is an internal function that consists in computing the hash of the received participation. In case of an equality of the computed hash and the received hash, the producer's participation is added to the group's participation set. When all the producers update their data or after a defined time, the chosen aggregator retrieves the group's participation set, computes the aggregated result, and updates the request result using the **updateRequestResult** function. Once this function's event is emitted, the consumer retrieves the result using the **getRequestResult** function. It decrypts the result with the appropriate group's secret key off-blockchain. The state of the consumer's group is *IN_PROGRESS* until its expiration time limit. After that, the group is ended and its state becomes *FINISHED* by the **endGroup** function.

Based on the proposed smart contract, we detail hereafter the dynamic aspect of the homomorphic encryption-based IoT data aggregation module by introducing both privacy policy generation and IoT data aggregation processes.

4.7.2 Privacy policy generation process

Figure 4.14 depicts the steps during the privacy policy generation process. We combine blockchain and off-blockchain semantic computations in order to generate and share a privacy policy between a consumer and a data producer. As mentioned above, through the semantic IoT gateway, the data owner uses the privacy preference manager to define some preferences about each IoT

data item collected by its data producers. Moreover, the blockchain client enables the semantic IoT gateway to interact with the public blockchain where consumers host their smart contracts. The privacy policy generation process begins when the consumer uploads its terms of service by invoking the **updateToS** function that is defined in the IoTDataAggregation smart contract. Once this function's event is emitted, each semantic IoT gateway computes a new privacy policy using its semantic rule manager. Based on our proposed set of inference rules and LIoPY ontology, the semantic rule manager matches the terms of service and the owner's privacy preferences, then returns back the result to the blockchain client of the semantic IoT gateway. In step four, a privacy policy is created only if all the privacy rule's privacy attributes match all the privacy attributes of the consumer's terms of service. A set of effective privacy attributes are associated with the generated privacy policy. Thus, the generated privacy policy defines how the data can be handled by the consumer once shared.

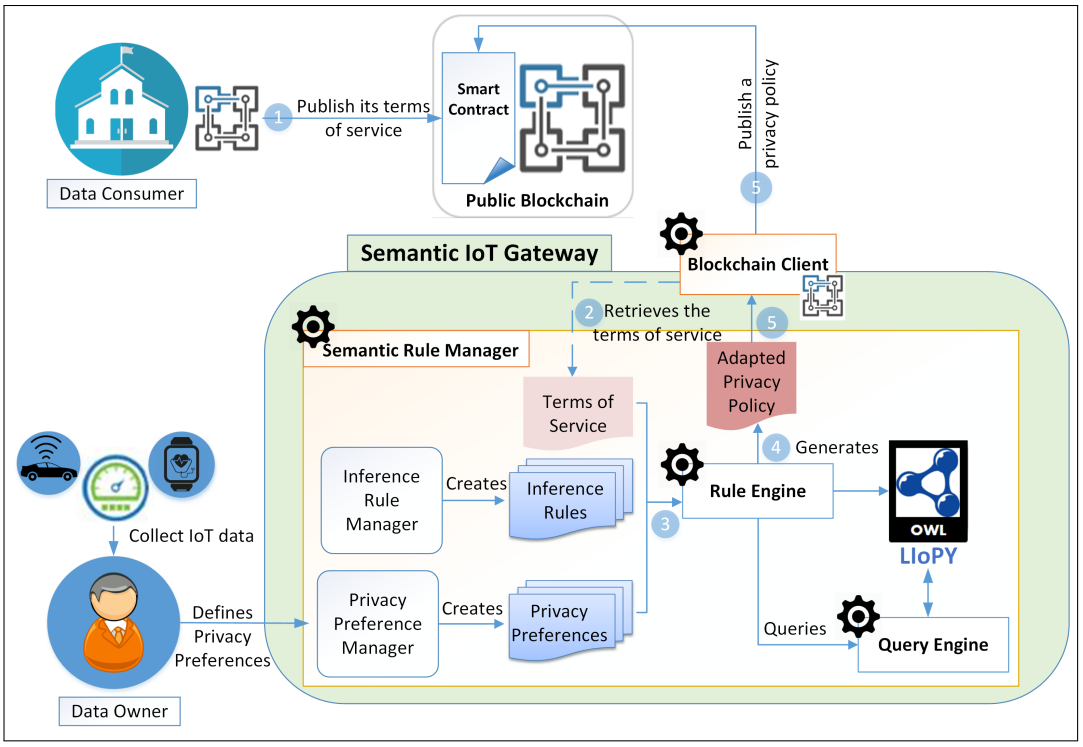


Figure 4.14: Privacy policy generation process

Finally, the semantic IoT gateway uploads the generated privacy policy on the smart contract by invoking the **updatePrivacyPolicy** function of the IoTDataAggregation smart contract. The reason behind storing the privacy policy on the smart contract has a twofold benefit. First, the smart contract can enforce the privacy policy compliance by preventing any privacy violation attempts without the involvement of a trusted third-party. Indeed, before adding any data producer to a specific group, the smart contract checks the requested data against the allowed and prohibited data items that are defined on the data producer's privacy policy. Second, the smart contract improves the non-repudiation principle compliance, which consists in preventing any IoT network node from denying actions that are performed by itself. Indeed, the producers'

owners cannot deny having sent their consent to use their IoT data. Nevertheless, the users can revoke the granted privacy permissions for a specific consumer by updating their consent to *Deny* instead of *Permit*.

Once each data producer has an associated privacy policy according to the consumer's terms of service, the IoT data aggregation process can begin.

4.7.3 IoT data aggregation process

Figure 4.15 depicts the different interactions between all the involved parties during the IoT data aggregation process. We can distinguish two processes, the first process depicted in the figure with blue lines and circles is when the semantic IoT gateway participates to the blockchain network as a group member that retrieves the appropriate public keys, encrypts the received ciphertext using the chosen aggregator's public key, and sends the result to the smart contract. Therefore, the second process depicted in the figure with green lines and rectangles introduces the different interactions when the semantic IoT gateway acts as the group's aggregator. Thus, it verifies the group's data, computes the aggregation, and publishes the result.

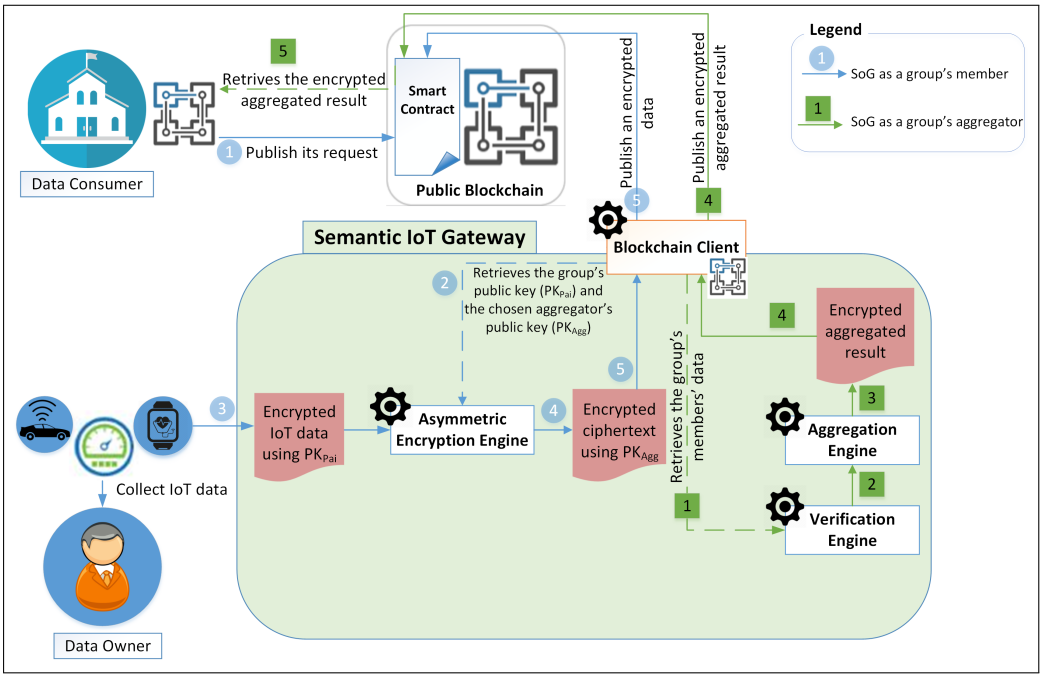


Figure 4.15: IoT data aggregation process

In order to detail the proposed processes, we discuss in the following six phases, namely system initialization, data collection, data transmission, data verification, data aggregation, and data aggregated reading.

4.7.3.1 System initialization

When the consumer wants to send a data request to the data producers (i.e., smart devices), it sends a transaction that invokes the smart contract constructor to host the smart contract

on the blockchain while indicating the blockchain address of the key generation node. The key generation node must be set when deploying/hosting the smart contract and remains immutable thus, it cannot be replaced.

Once the smart contract is hosted on the blockchain, the consumer defines its terms of service, chooses the requested IoT data, and creates a group whose role is to produce an aggregated result while complicating sensitive information inference from single data participant. On the other hand, the smart contract enables the data producers to introduce the privacy choices of their owners. Thus, based on these privacy choices, the smart contract can decide whether or not one producer will be included in a specific group. Moreover, an aggregator is randomly chosen to aggregate the data of all the group members and update the result on the smart contract. The request result update function can only be invoked by the aggregator's blockchain address.

Then, for each created group, the key generation node chooses two large primes p and q. Then, it runs the key generation function of the Paillier cryptosystem [Paillier, 1999] to obtain a public key, denoted as $Pk_{Pai} = (N, g)$, with $N = pq$ and the corresponding private key, denoted as $Sk_{Pai} = (\lambda(N), \mu)$.

After that, the key generation node sends a transaction that invokes a smart contract function to update the group's public key with the generated one i.e., Pk_{Pai}, which will be accessible for the group members. The group's public key update function can only be invoked by the key generation node's blockchain address. The private key Sk_{Pai} is only shared with the consumer (i.e., the smart contract owner).

To sum up, the system initialization phase consists in creating a consumer's smart contract, grouping the data producers based on their privacy policies, and sharing the generated keys with the data producers and the consumer.

4.7.3.2 Data collection

Based on the privacy policies, the smart contract decides whether or not one smart device is included in the created group. In case of an inclusion, each group member collects its produced data according to the requested data in the appropriate group. Then, it retrieves the group's public key of the Paillier cryptosystem [Paillier, 1999] published by the key generation node and encrypts the data.

Indeed, let a group with k smart devices. Each smart device SD_i, with $1 \leq i \leq k$ collects periodically data, denoted as M_i, randomly chooses a number such that $r \in Z_N^*$, and computes the encrypted data, denoted as $participation_i$ using the public key $Pk_{Pai} = (N, g)$ as follows:

$$participation_i = Enc(M_i, Pk_{Pai}) = g^{M_i} r^N \mod N^2 \tag{4.4}$$

Both the encrypted collected data *participation* and the public key Pk_{Pai} are the main elements used in the rest of the IoT data lifecycle, namely the data transmission, verification, aggregation, until data aggregated reading.

4.7.3.3 Data transmission

In order to secure the data during the transmission phase, encryption, hash, and digital signature techniques are applied.

In order to guarantee the data confidentiality, the chosen aggregator shares a public key, denoted as Pk_{Agg} with the group members and keeps the corresponding private key, denoted as Sk_{Agg} secret. Each group member encrypts the collected data, denoted as $participation_i$ with the aggregator's public key such that:

$$encParticipation_i = Enc\big(participation_i, Pk_{Agg}\big) \tag{4.5}$$

Moreover, the hash functionality and the digital signature are used in order to enable the message's receiver (i.e., the chosen aggregator) to check the data integrity and verify the sender's identity. Each participant of the blockchain network has at least a blockchain address with one related couple of keys, a public key Pk_{SD} that is shared by all participants and a private key Sk_{SD} that is kept secret by each participant.

The *hash* is calculated from the encrypted produced data, denoted as *encParticipation* using the Secure Hash Algorithm SHA-2(256) [Sklavos and Koufopavlou, 2003] such that:

$$hash_{encP} = SHA256\big(encParticipation\big) \tag{4.6}$$

The digital signature, denoted as *signature* refers to the ciphertext of the digest produced by the sender's private key Sk_{SD}.

$$signature_{hash} = Sign\Big(SHA256\big(encParticipation\big), Sk_{SD}\Big) \tag{4.7}$$

Both the data hash *hash* and the digital signature *signature* are sent with the encrypted data *encParticipation* to be stored on the smart contract. This latter offers two verification methods, that consist of the hash value verification and the signature verification.

4.7.3.4 Data verification

To verify the transmitted data, the smart contract computes the *computed_hash*, that is the hash of the received encrypted data, denoted as *received_encParticipation* using the hash function *keccak*256 such that:

$$computed\_hash = keccak256\big(received\_encParticipation\big) \tag{4.8}$$

In case of an equality of both the received hash, denoted as *hash* and the computed hash, denoted as *computed_hash*, the smart device's participation is accepted and added to the group's participation list.

Moreover, the smart contract enables the aggregator to verify the smart device's identity, denoted as *sender_identity* that presents the sender's blockchain address. Hence, the signature

verification function, denoted as *ecrecover* recovers the blockchain address using the hash of the encrypted data and its signature such that:

$$computed\_identity = ecrecover\big(hash, signature\big) \tag{4.9}$$

The *computed_identity* is equal to the sender's identity only if the same private key is used to sign the hash data.

In sum, the data's integrity and the sender's identity are verified only if the *computed_hash* is equal to the received *hash* and the *computed_identity* is equal to the public *sender_identity*. Both data verification functionality is offered by the proposed smart contract.

4.7.3.5 Data aggregation

When the chosen aggregator retrieves the participations of all the group members, it verifies the sender's identity and the data's integrity in order to prevent illegal smart devices sending malicious data. If the data are verified, it decrypts all the participations using its private key, denoted as Sk_{Agg}. After that, the aggregator computes the sum of all the data, denoted as M_i, with $1 \leq i \leq k$ and k is the number of the group members as follows:

$$\begin{aligned} encRequestResult &= Enc\left(\sum_{i=1}^{k} M_i, Pk_{Pai}\right) \\ &= \prod_{i=1}^{k} Enc\big(M_i, Pk_{Pai}\big) \\ &= \prod_{i=1}^{k} g^{M_i} r^N \mod N^2 \\ &= g^{\sum_{i=1}^{k} M_i} \left(\prod_{i=1}^{k} r_i\right)^N \mod N^2 \end{aligned} \tag{4.10}$$

After computing the aggregation data, the aggregator that was randomly chosen for a specific group request updates the request result by sending a transaction to the smart contract. When the request result is stored on the smart contract, the *requestResultUpdated* event is emitted. This event has two parameters, namely the group ID and the request's result that is the encrypted aggregated data.

4.7.3.6 Data aggregated reading

Once the result is updated on the smart contract, the consumer retrieves the appropriate group's request result, denoted as *encRequestResult*. Then, the consumer uses the private key $Sk_{Pai} =$

$(\lambda(N), \mu)$ of the Paillier cryptosystem [Paillier, 1999] to decrypt the request result such that:

$$\begin{aligned} request\_result &= Dec\left(\frac{L\left(encRequestResult^{\lambda} \mod N^2\right)}{L\left(g^{\lambda} \mod N^2\right)} \mod N, Sk_{Pai}\right) \\ &= \sum_{i=1}^{k} M_i \end{aligned} \tag{4.11}$$

After that, the consumer ends the group by invoking the endGroup smart contract function. Otherwise, the result is rejected.

In this way, the consumer can read the sum of all the group members without knowing the individual IoT data.

4.8 Summary

The blockchain technology is a distributed database that records all the transactions that have ever occurred in the network. The main feature of the blockchain is that it allows deploying smart contracts. In fact, a smart contract is an executable code that runs on top of the blockchain to facilitate, execute and enforce an agreement between untrusted parties without the involvement of a trusted third-party. Hosting a smart contract in the blockchain can enforce privacy-preserving in the IoT domain. For this purpose, we have proposed an end-to-end privacy-preserving framework for the IoT data based on the blockchain technology, called PrivBlockchain. This latter consists of three networks, namely private area, regional area, and blockchain. In order to ensure the three networks interaction, we defined a Semantic IoT Gateway that acts as a bridge between the IoT devices and the three aforementioned networks. The main functionality of our proposed PrivBlockchain framework are: first, enforcing the data owner's privacy preferences on how the own IoT devices must behave according to each data output. Second, matching the owner's preferences and the consumer's requirements in order to infer a privacy policy using LIoPY, a European Legal compliant ontology for supporting preserving IoT PrivacY (see Chapter 2) as well as a set of inference rules (see Chapter 3). Third, converting the inferred privacy policy into a custom smart contract using a set of predefined set of functions. Indeed, the use of smart contract aims at enforcing the privacy requirements when sharing the IoT data. Finally, using the homomorphic encryption technology in order to aggregate IoT data at group-level and tackle the raw data disclosure issue.

In the next chapter, we implement our defined smart contracts, validate them in a blockchain test network, and conclude with an analysis.

Chapter 5

Evaluation and Analysis

Table of Contents

5.1 Introduction . . . 121
5.2 Evaluation . . . 122
5.2.1 Motivating scenarios . . . 122
5.2.2 Experimental environment . . . 123
5.3 Blockchain-based smart home . . . 123
5.3.1 IoT device management use case . . . 123
5.3.2 Security and privacy analysis . . . 127
5.3.3 Performance Evaluation . . . 128
5.4 Blockchain-based healthcare system . . . 130
5.4.1 IoT data sharing use case . . . 130
5.4.2 Security and privacy analysis . . . 134
5.4.3 Performance Evaluation . . . 135
5.5 Blockchain-based smart grid . . . 138
5.5.1 IoT data aggregation use case . . . 139
5.5.2 Security and privacy analysis . . . 140
5.5.3 Performance Evaluation . . . 141
5.6 Comparative performance analysis . . . 144
5.7 Summary: privacy design strategies compliance . . . 149

5.1 Introduction

In the previous chapter, we present the theoretical framework. In this chapter, we implement the defined smart contracts and validate them in a blockchain test network. Our evaluation aims at (i) proving that the proposed model is implementable, (ii) providing a performance analysis in terms of processing time, scalability, and cost per transaction, and (iii) deciding whether our

solution can be used in practice or it is cost-expensive. Therefore, we first introduce a general overview of our evaluation by defining three motivating scenarios and the chosen experimental environment. Then, we detail each scenario in a separate section while presenting a use case, security and privacy analysis, and performance evaluation.

This chapter is organized as follows. Section 5.2 presents a general overview of our evaluation. Section 5.3 defines our blockchain-based smart home. Section 5.4 defines our blockchain-based healthcare system. Section 5.5 defines our blockchain-based smart grid. Section 5.6 analyzes the proposed use cases before summarizing the content of this chapter in Section 5.7.

5.2 Evaluation

In this section, we first define three motivating scenarios, then we present the chosen experimental environment used to implement our test systems.

5.2.1 Motivating scenarios

We describe below three motivating scenarios including smart home, healthcare, and smart grid domains:

- **Smart home domain**: Emma, a 30-year old woman who needs to follow a healthcare protocol, which consists in practicing some sport activities and eating healthy meals. To this end, she uses some wireless body sensors, which measure vital parameters, such as heartbeat and blood pressure as well as sport activities parameters, such as steps and training duration. The sensors collect Emma's data and send them to the home-gateway that decides what information to send to the hospital to be stored on Emma's medical base, which is regularly checked by her doctor. Emma is afraid of losing the control over her smart devices. Thus, she looks for a solution that allows her to define her privacy permission settings for each device and guarantees their enforcement.

- **Healthcare domain**: Bob is a patient in a hospital that provides e-medical care to its patients. Thus, Bob has access to everything related to his health information, namely doctor's notes about the consultations, blood test results, and prescribed medications through a Patient Portal. Besides, Bob can associate some IoT wearable devices that collect the patient's location and vital signs and send them to the hospital. Thus, Bob's location, heartbeat, and breath rate can be accessible by the emergency service crew in case of an emergency situation for Bob. Bob looks for a solution that allows him to have an overview of everything related to how his data are handled and define his privacy preferences about who can/cannot access his data.

- **Smart grid domain**: lets us consider a set of smart meters installed at homes that are connected to a few servers under the control of an energy substation. All these nodes communicate together through a network set up and maintained by the energy substation. Smart meters collect and periodically store energy consumption data. Afterwards, data will

be transferred to the energy substation. Then, data will be used for forecasting algorithms, fraud detection, and fault prevention or diagnosis. However, sending all the produced data to the energy substation is the worst privacy choice. Privacy should be preserved before the transmission phase instead of trying to preserve it when the data are already stored in the substation data center. Indeed, the smart meters' owners look for a solution that allows them to collaborate among them and aggregate their data in order to enhance their privacy.

5.2.2 Experimental environment

In the previous chapter, we propose a set of smart contracts. In order to validate them, we use Ethereum, which is currently the most common blockchain platform for developing smart contracts [Buterin et al., 2014]. Hence, we implement our smart contracts using the Solidity language [Solidity, 2014] and deploy them to the Ethereum test network. Because our system dœs not rely on the currency transfer, there is no difference between the real Ethereum network and the Ethereum test network. Thus, we create three test systems using Truffle development framework [Truffle, 2016], which is the most popular development framework for Ethereum. This framework, among others, generates JavaScript bindings for the smart contract, enables automated smart contract testing, and includes libraries such as web3.js [Web3, 2017] that facilitates the communication between the smart contract and Ethereum clients. During our experiments, we use the contract events in order to automate the actions taken by the different nodes. Thus, we implement event callbacks in our testing framework using the web3.js library [Web3, 2017].

5.3 Blockchain-based smart home

In this section, we define a use case for IoT device management, analyze the security and privacy properties, and evaluate the performance of the proposed blockchain-based smart home solution.

5.3.1 IoT device management use case

We implement a test system that consists of several nodes, namely 1 data owner, 1 semantic IoT gateway, and 50 smart devices. We assume that Emma's smart devices are connected to the PrivBlockchain framework and each of them is identified by a blockchain address.

Let Emma a data owner that has a set of smart devices that help her to follow a healthcare protocol, which consists in practicing some sport activities and eating healthy meals. Let the smart devices as one wearable sensor, one smart treadmill, and a smart phone. Let the semantic IoT gateway as one tablet computer that has high memory and storage capabilities. Emma owns these smart devices that collect her heartbeat, steps, and training duration. Figure 5.1 depicts an example of our test system's screen shot during the privacy permission setting definition. First, Emma deploys the BehaviorControl smart contract using her tablet computer. Second, she updates the privacy permission settings of each IoT device using the **privacySettingAdd** function. For instance, Emma allows (i) the wearable device to locally store her heartbeat on the gateway

as shown in Figure 5.1 by the first emitted event, (ii) the smart treadmill to collect her steps and monitor her heartbeat as shown in Figure 5.1 by the second and third emitted events, and (iii) the semantic IoT gateway to externalize the collected data as shown in Figure 5.1 by the last emitted event. Indeed, during the training, the wearable device collects Emma's vital parameters and sends them to her semantic IoT gateway. This latter receives Emma's sensitive data and can send them to the hospital to be stored on Emma's medical base, which is regularly checked by her doctor. Moreover, these stored data are analyzed to propose personalized recommendations for users. Hence, a need for a break or water notifications can be sent to Emma when necessary.

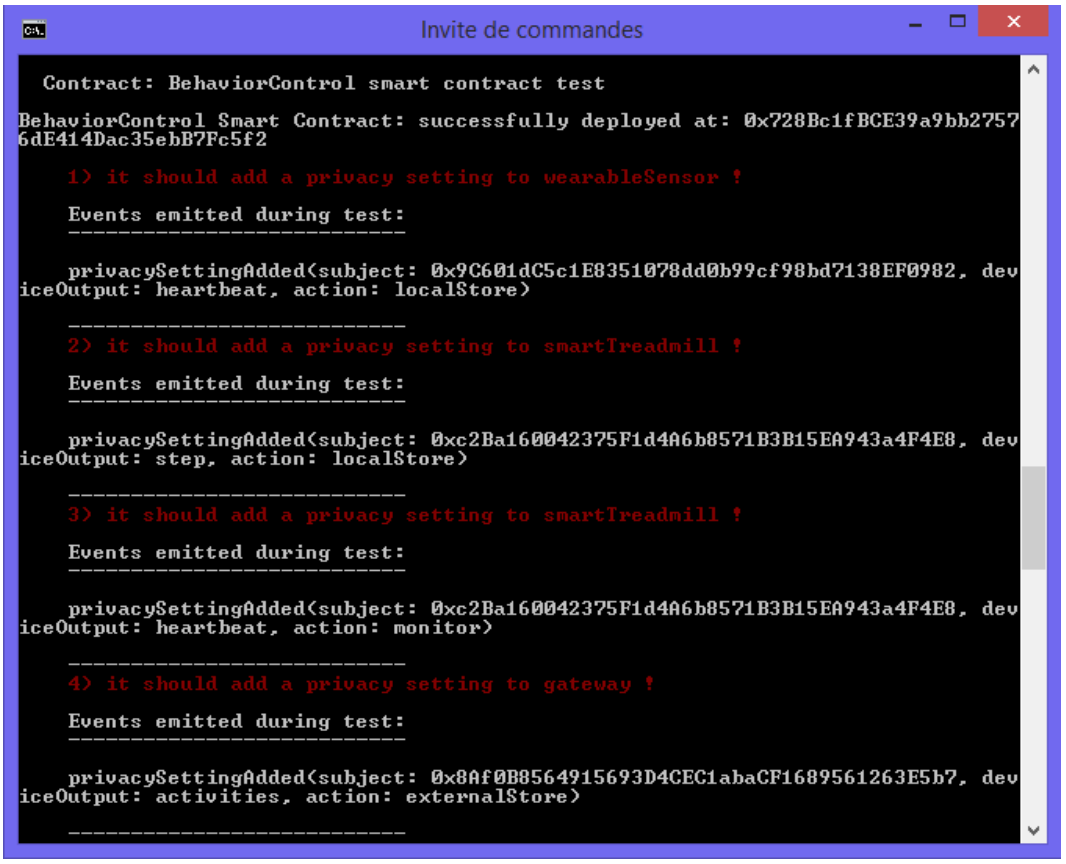


Figure 5.1: Blockchain-based smart home test system screenshot during privacy permission setting definition

Figure 5.2 depicts an example of our test system's screen shot during the privacy permission setting verification. Each smart device invokes the appropriate function defined on the PrivacyPermissionSetting smart contract to be authorized to execute the needed operation. For instance, the wearable device invokes the **LocalStore** function defined on the PrivacyPermissionSetting smart contract to be able to store the produced heartbeat data on Emma's gateway. Indeed, in order to enforce the privacy permission settings of the wearable device, the **verifyPermission** function is executed in order to allow the requested actions. Thus, this function first verifies the smart device's authorizations, then analyzes the smart device's behavior, and finally emits the **ReturnRequestResult** event with the appropriate decision, which is 'Action Authorized !' in our case as shown in Figure 5.2 by the last emitted event.

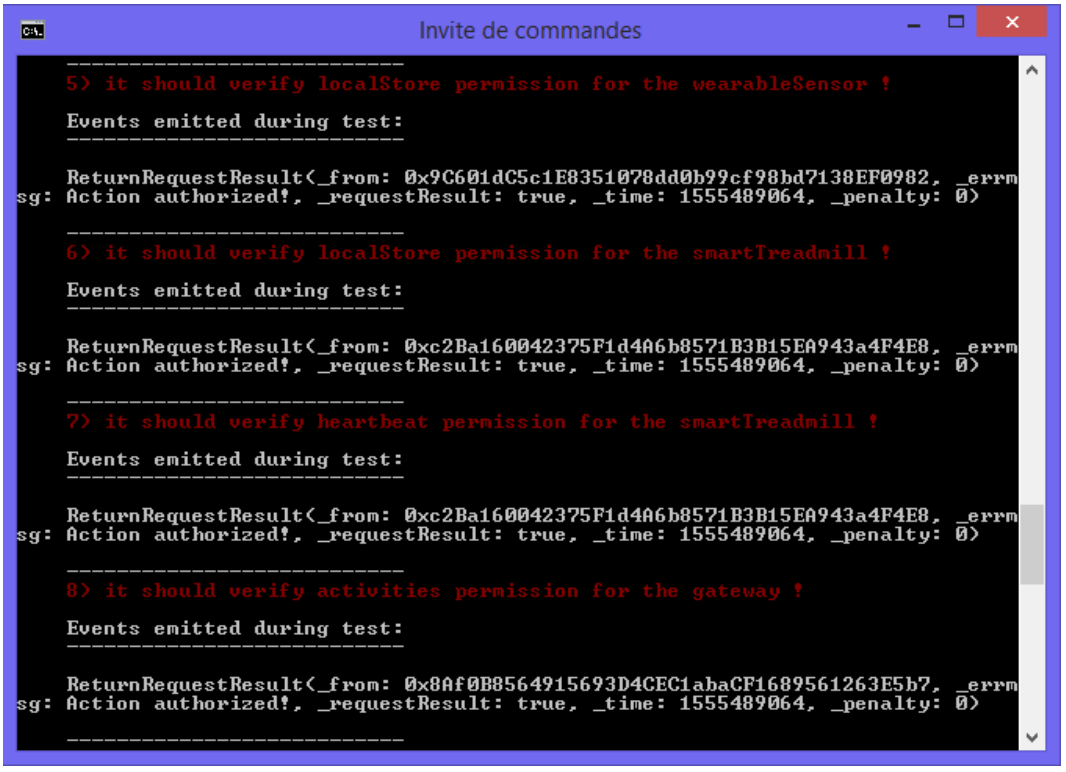


Figure 5.2: Blockchain-based smart home test system screenshot during privacy permission setting verification

Let a Denial of Service (DoS) attack in which an attacker sends a lot of transactions to the same target in a very short time. We conduct two experiments to simulate this kind of attack. The first experiment consists in sending a lot of transactions to the same target using one blockchain address. The second one consists in sending a lot of transactions to the same target using several blockchain addresses.

Figure 5.3 depicts the result of the first experiment during the privacy permission setting violation attempts. Let a wearable sensor that sends several access requests to the heartbeat resource using its blockchain address. The BehaviorControl smart contract authorizes the action, then detects a misbehavior, and blocks that address for few minutes. The penalty (i.e., the block duration) is computed according to the detected misbehavior number in the past. During block time, the wearable sensor cannot access to the heartbeat resource, whereas other sensors like smart treadmill can access to the heartbeat.

Figure 5.4 depicts the result of the second experiment. Let several blockchain addresses that send several access requests to one resource, such that heartbeat resource in our example. The BehaviorControl smart contract detects this misbehavior and blocks the access to that target to protect it.

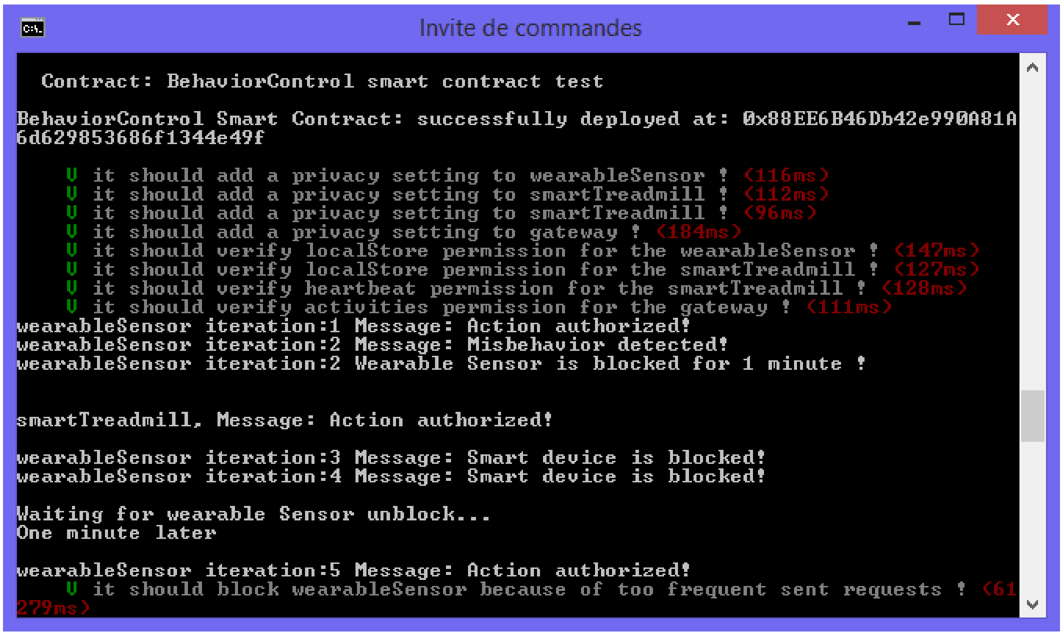


Figure 5.3: Blockchain-based smart home test system screenshot in case of denial of service from one blockchain address

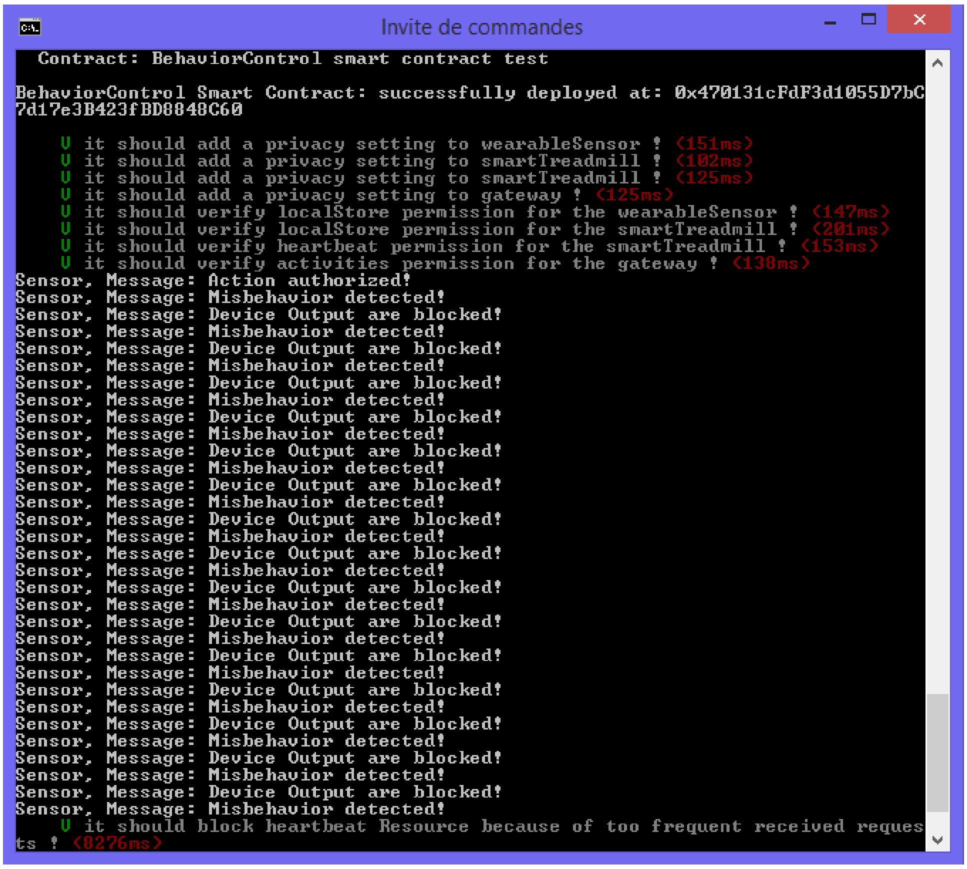


Figure 5.4: Blockchain-based smart home test system screenshot in case of denial of service from several blockchain addresses

5.3.2 Security and privacy analysis

After detailing the IoT device management test system, we highlight and analyze in this section both security and privacy properties.

5.3.2.1 Anonymity and pseudonymity

Each smart device has a blockchain address used to communicate with other devices. Thus, the anonymity aims at tying the smart devices in order to obfuscate the data owner's habits and personal behaviors.

To break anonymity, an attacker may try to link anonymous transactions and other available information to find the data owner's real identity. In order to protect against such linking attack, the blockchain addresses of all the smart devices are periodically updated. Indeed, by using different pseudonyms, an attacker is prevented to link real world identities and pseudonyms.

5.3.2.2 Authentication and privacy permission setting control

Each smart device has a blockchain address and a set of privacy permission settings, which defines how each smart device must behave, such that where store its produced data, which other devices can communicate with, and with which frequency per seconds for each operation.

In order to enforce the privacy permission settings, each smart device has a PrivacyPermissionSetting smart contract that includes the authorized operations defined according to the privacy preferences of the data owner. For instance, a smart device without external store permission dœs not have the appropriate function on its associated smart contract. Thus, such solution can be considered as a preventive one.

To break authentication and smart device control, an attacker may take control of one smart device and start to use the predefined functions on the smart contract to attack the network. In order to address this attack, our design employs behavior monitoring that detects smart devices' misbehavior thanks to the BehaviorControl smart contract. Moreover, the semantic IoT gateway controls all transactions in the network. In order to protect the smart devices from malicious requests, the transactions are filtered and limited to the authorized transactions by the BehaviorControl smart contract. Thus, the semantic IoT gateway forwards only the requests sent by the accepted transactions to the devices to be executed.

Furthermore, only the data owner blockchain address can update the privacy permission settings of the owned smart devices. Thus, the semantic IoT gateway only executes the smart contract code but cannot modify it or alter the smart devices authorizations.

5.3.2.3 Availability

Each smart device or IoT resource (i.e., produced data) are available to legitimate users. Thus, the availability means that the target is accessible when it is needed.

To break availability, an attacker may take control of one smart device and send multiple transactions to one IoT resource. In order to protect against such denial of service attack, the

BehaviorControl smart contract hosted on the blockchain detects smart devices' misbehavior and blocks their blockchain addresses.

5.3.3 Performance Evaluation

In this section, our proposed smart home system's performance is evaluated in terms of computation time cost and scalability overhead.

5.3.3.1 Computation time cost

In order to evaluate the performance of our solution, we conduct an experiment to compute the processing time needed by one semantic IoT gateway to validate a privacy permission setting definition transaction that invokes the **privacySettingAdd** function and a privacy permission setting verification transaction that invokes the **verifyPermission** function defined on the BehaviorControl smart contract. First, we conduct an experiment to measure the processing time of invoking both **privacySettingAdd** and **verifyPermission** functions. Figure 5.5 depicts the computational cost of the two functions for one smart device. Only 150 milliseconds is needed in order to add a new privacy permission setting or verify the smart device's behavior for one smart device.

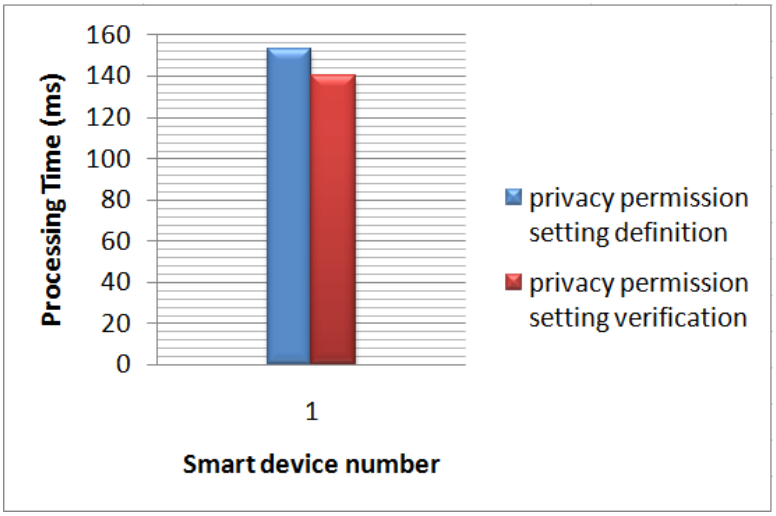


Figure 5.5: Computational cost of privacy permission setting definition and verification for one smart device

After that, we conduct the same experiment while increasing the number of smart devices managed by one semantic IoT gateway. Figure 5.6 depicts the computational cost of the two functions while increasing the smart device number from 1 to 5. The processing time varies from 150 to 750 ms. We observe that the processing time is equal to the processing time for one smart device multiply by the smart device number. Thus, the more the smart device number increases, the more the gateway's computing capabilities are required in order to reduce the processing time.

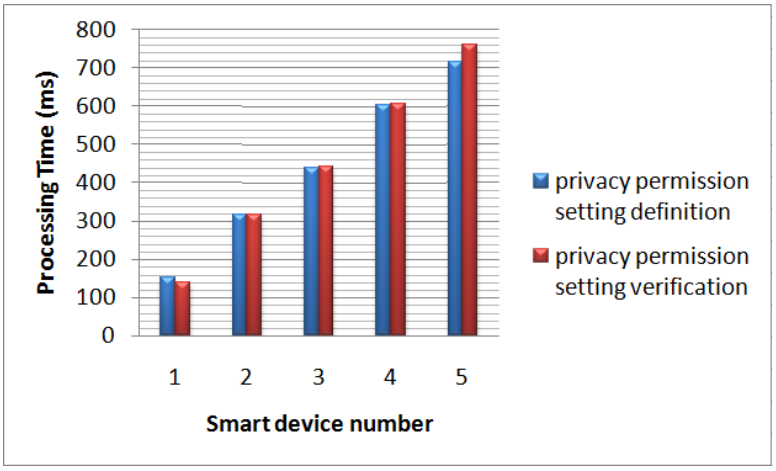


Figure 5.6: Average computational cost of privacy permission setting definition and verification for five smart devices

5.3.3.2 Scalability overhead

In order to evaluate the scalability of both the **privacySettingAdd** and **verifyPermission** functions, we make several tests while increasing the number of the managed smart devices by the gateway from 1 to 50. We run the simulation for 60 seconds during which a total of 554 transactions are created. Figure 5.7 depicts the average of 10 runs of the simulation. We observe that the processing time increases with the number of smart devices, going from 100 to 8000 ms. Thus, one semantic IoT gateway can manage 50 smart devices in about 8 seconds, which is a short delay time while improving the data owner control over the owned smart devices.

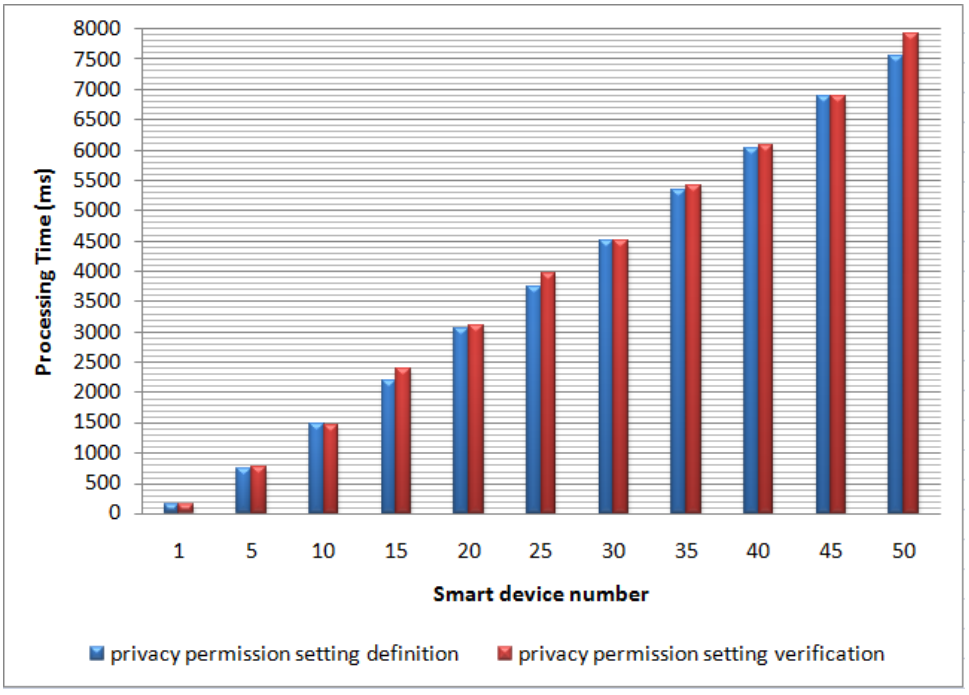


Figure 5.7: Average computational cost of privacy permission setting definition and verification for fifty smart devices

5.4 Blockchain-based healthcare system

In this section, we define a use case for IoT data sharing, analyze the security and privacy properties, and evaluate the performance of the proposed blockchain-based healthcare system.

5.4.1 IoT data sharing use case

We implement a test system that consists of several nodes, namely 1 patient, 1 family doctor, 1 emergency service, 1 emergency service dispatcher, 1 ambulance crew, 1 emergency doctor, and 1 pharmacist. We assume that each node has a blockchain address.

Suppose a hospital that provides e-medical care to its patients and the healthcare providers, such as doctors, pharmacists, and emergency service crew. Let *e-Medical records* a digital register that includes several medical information about a patient, such that blood test results, patient's treatment, and e-consultations with the family physician and medical specialists. Besides, it includes a medical prescription digital file, called *e-Prescription*. The medical prescription is an authorization written by a doctor or other qualified healthcare practitioner for a patient to purchase a prescription drug from a pharmacist. Let *e-Prescription* a digital file accessed by three actors with different permissions. Thus, the doctor is allowed to read and write on this file, whereas the patient and the pharmacist can only read the *e-Prescription* digital file. Moreover, let *e-Emergency medical records* a second digital register that includes several information about a patient, namely age, blood group, and allergies. Besides, other near real-time data can be added by some IoT wearable devices that collect the patient's location and vital signs, such as heartbeat, breath rate, and blood pressure. This digital register is accessible by the emergency service crew in case of an emergency situation for the patient. The emergency service crew includes an emergency service dispatcher, an ambulance crew, and hospital's emergency doctors. Both ambulance crew and hospital's emergency doctors can add new digital files to the *e-Emergency medical records* register, namely *e-Ambulance report* and *e-Prescription*, respectively.

Figure 5.8 depicts an example of our test system's screen shot during the registration phase. Let Bob a patient that grants the access authorization to his family doctor and the emergency service to access his *e-Emergency medical records* register. First, Bob deploys the IoTDataSharing smart contract while introducing his family doctor's blockchain address. Second, the family doctor adds a new digital file, called *e-GeneralRecords* that includes some patient's information, namely age, blood group, allergies, and patient's vital signs using the **addFile** function (see Event 1>). After that, the emergency service updates its terms of service by invoking the **updateToS** function defined on the smart contract (see Event 2>). According to the emergency service's terms of service, a common privacy policy that matches Bob's privacy preferences is generated. The privacy policy generation code is a JAR (Java ARchive) file, which is a package file format used to store many Java classes and associated metadata into one file for distribution. The used JAR is based on Algorithm 1 (defined on Chapter 3) that consists in matching the owner's privacy preferences and the consumer's terms of service in order to generate a common privacy policy. For instance, when the privacy policy specifies that the consumer has the *disclosure per-*

mission this is translated on the smart contract by the consumer's ability to invoke **addConsumer** function with limited retention and disclosure permissions for the new one. Once the privacy policy is generated, Bob adds the emergency service's blockchain address as a consumer to the *e-Emergency medical records* register's smart contract using the **addConsumer** function and the generated set of privacy permissions as this function's parameters (see Event 3>). Indeed, the emergency service is not allowed to write on the *e-GeneralRecords* file but can disclosure it.

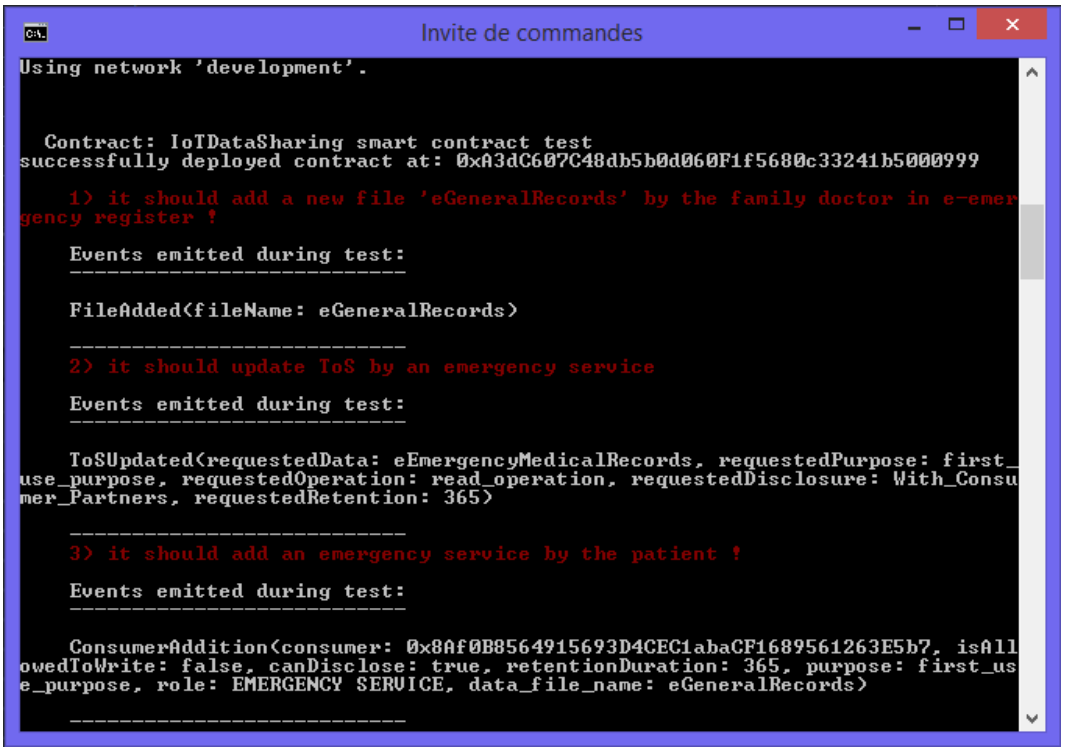


Figure 5.8: Blockchain-based healthcare test system screenshot during registration time

Figure 5.9 depicts an example of our test system's screen shot in case of an emergency situation. Let Bob has a traffic accident that causes bleeding on Bob's arm. Alice, his wife calls the hospital emergency service and give them Bob's digital identity. Thus, the service emergency uses the **addConsumer** function in order to add an emergency service dispatcher as a consumer with *disclosure permission* on the *e-GeneralRecords* digital file. This dispatcher accesses to Bob's *e-Emergency medical records* register as a consumer and disclosed Bob's register to the ambulance car that is rushing to the traffic accident location. Indeed, the dispatcher uses the **addConsumer** function in order to add the ambulance crew on Bob's emergency smart contract using its disclosure permission. Thus, the ambulance crew have access to the patient's emergency medical records. Let Bob has haemophilia, which means that his blood dœs not clot. This information allows for the right decisions to be made quickly by the ambulance crew.

On their way to the hospital, the ambulance crew assess Bob's condition on a *e-Ambulance report* digital file using the **addFile** function and update the patient's *e-Emergency medical records* digital register, which has also been accessible by the hospital's emergency doctors allowing to rapidly make any preparations before the patient's arrival.

For Bob, the accident was not too serious. Thus, the hospital's emergency doctor prescribes a list of medications to Bob's by creating a new *e-Prescription* digital file using the **addFile** function. Then, the emergency doctor adds the patient as a consumer that can read and disclosure the *e-Prescription* digital file using the **addConsumer** function. After that, Bob grants the access authorization to his pharmacist for one day, then go to the pharmacy to buy medications. The pharmacist uses the digital prescriptions, which means that Bob only has to provide his digital identity. Thus, the pharmacist can access the *e-Prescription* file to see the prescribed medications.

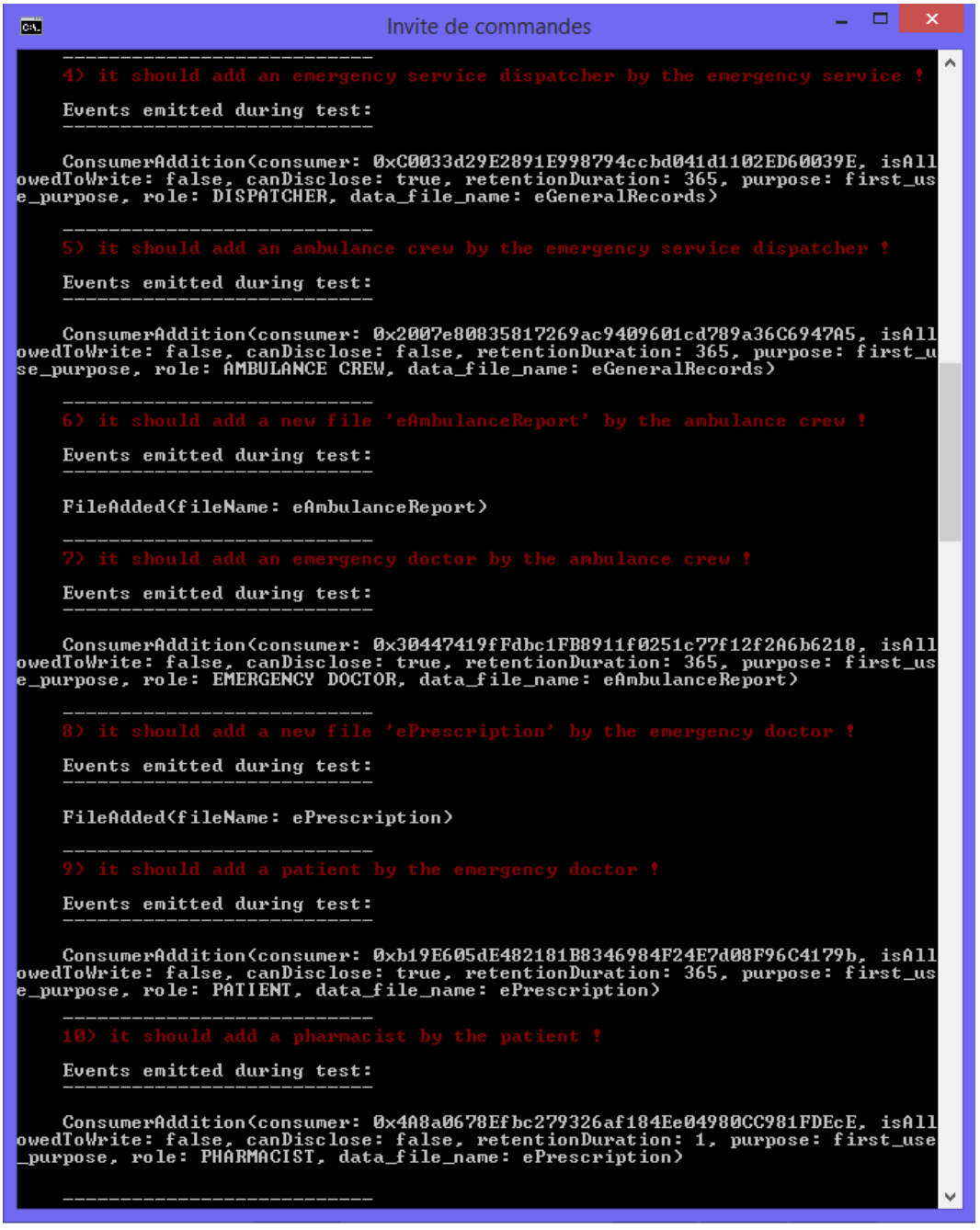


Figure 5.9: Blockchain-based healthcare test system screenshot in case of an emergency

Figure 5.10 depicts some detected privacy violation attempts. For instance, Bob is not allowed to write on the *e-Prescription* file. In case of an attempt to save any modification on this file, our smart contract rejects such attempt and logs it on the blockchain. Since Bob dœs not authorize the pharmacist to disclosure the *e-Prescription* file (i.e., *canDisclosure: false*), a disclosure attempt notification is emitted when the pharmacist tries to add some consumers to the shared file.

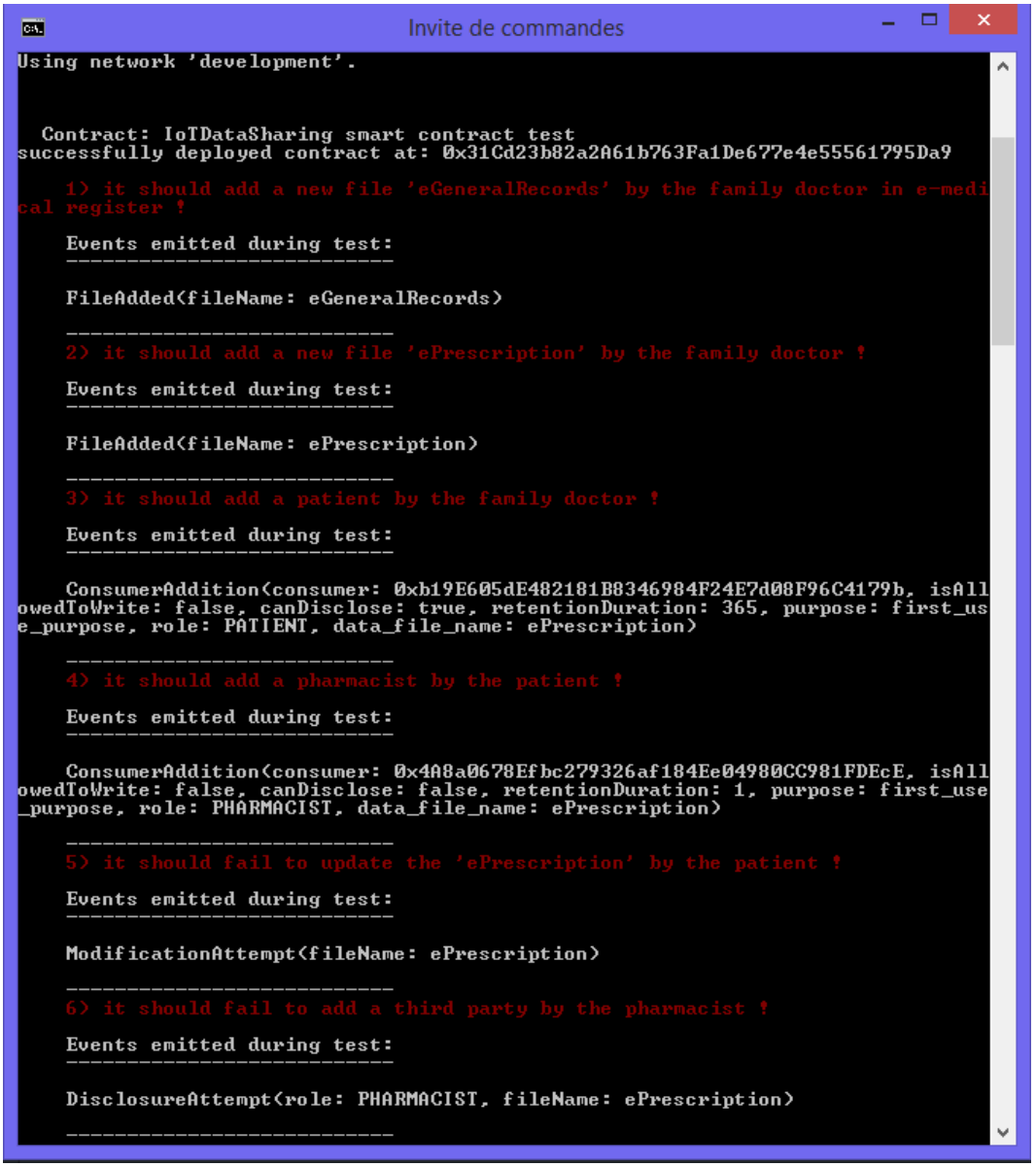


Figure 5.10: Blockchain-based healthcare test system screenshot in case of privacy violation attempts

The hospital's healthcare providers are allowed to access the medical care though the patient can always revoke this right using the **removeConsumer** function defined on the IoTDataSharing smart contract. Every query made about the patient is logged, which makes the system reliable and traceable.

5.4.2 Security and privacy analysis

After detailing the IoT data sharing test system, we highlight and analyze in this section both the security and privacy properties.

5.4.2.1 Anonymity and pseudonymity

By creating for each patient a digital identity defined by a blockchain address, the real identity of a specific patient becomes obfuscated. Thus, pseudonymity aims at tying the patient's identity while participating in the blockchain network.

However, through analysis of the network traffic, some patterns of treatment, such that repeatedly interaction between two network entities can be inferred. In order to improve obfuscation while preserving auditability on the blockchain, a permissioned blockchain can be used. Indeed, only healthcare providers, such as hospitals, pharmacies, and medical centers are allowed to access to the ledger. This prevents rogue actors from extracting frequency-based insights from the blockchain transactions. Furthermore, encryption is introduced in the off-blockchain data store to safeguard against accidental or malicious content access.

5.4.2.2 Data integrity and sender's identity

The proposed module can guarantee both data integrity and sender's identity checking.

For the data integrity, after each modification on the off-blockchain stored file content, a new hash is calculated, then the **updateFile** function is invoked in order to audit data manipulation. Thus, any file content manipulation can be detected by comparing the stored file content's hash and the computed file content's hash.

For the sender's identity checking, all the receivers can verify the transaction's sender's identity using the digital signature. Thus, any illegal transaction can be detected by comparing the sender's blockchain address and the recovered identity using the transaction's signature.

5.4.2.3 Blockchain-based healthcare system in legislation context: GDPR compliance

The proposed healthcare system aims at achieving the GDPR compliance by meeting its privacy requirements.

First, the GDPR specifies that the *'personal data should be processed on the basis of the consent of the data subject concerned'* [GDPR, 2016]. PrivBlockchain's module meets this requirement by using the IoTDataSharing smart contract to offer to the users the ability to manage their consent for sharing data with healthcare providers. Indeed, the user can easily add, modify, and revoke authorizations.

In addition to the user's consent management, our solution establishes accountability and transparency in the data exchange process between all the involved parties. Thus, the defined blockchain-based solution helps healthcare providers to automate compliance checks and allows for a comprehensible record for auditing.

Moreover, the GDPR specifies that the consumers should ensure *'preventing unauthorized access to or use of personal data'* [GDPR, 2016]. Using the IoTDataSharing smart contract turns the user's privacy choices into computer-readable set of rights and obligations. Thus, only added consumers to a specific file can access to its content. Moreover, any privacy violation attempts can be detected. For instance, the user is notified in case of an attempt to disclosure the file by an unauthorized consumer. Besides, by logging all the transactions, our solution proves who has accessed personal data, where, and when.

Furthermore, the GDPR specifies that *'in order to ensure that the personal data are not kept longer than necessary, time limits should be established by the controller for erasure'* [GDPR, 2016]. PrivBlockchain's module meets this requirement by using the off-blockchain data store and only storing the hash of the data on the blockchain. Thus, the user has the right to be forgotten and can ask to delete the shared data when the data retention duration is ended. For instance, the proposed smart contract can trigger a secure deletion of specific data.

By meeting the aforementioned privacy requirements, PrivBlockchain's module addresses areas associated with GDPR compliance. On one hand, it enforces the user's ownership and control over the shared data. On the other hand, it can be seen as a proof of legislation compliance by the consumers thanks to both transparency and auditability characteristics.

5.4.3 Performance Evaluation

In this section, our proposed healthcare system's performance is evaluated in terms of computation time cost and cost overhead.

5.4.3.1 Computation time cost

In order to measure the performance of our solution, we conduct some experiments to compute the computational time cost of some functions defined on the IoTDataSharing smart contract. We use add, update, share, and revoke file operations as a use case for our performance evaluation.

First, we conduct an experiment to measure the computation time cost of invoking four functions, namely **addFile**, **updateFile**, **addConsumer**, and **removeConsumer**. Figure 5.11 shows the average processing time of the function invocations. The processing time varies from 115 to 295 ms. Both **addConsumer** and **removeConsumer** functions are independent of the file size. However, for both **addFile** and **updateFile** functions, the processing time includes the hash computing time of the file content and the execution time of the function. For this purpose, we detail below these two functions while increasing the file size from 1KB to 2MB.

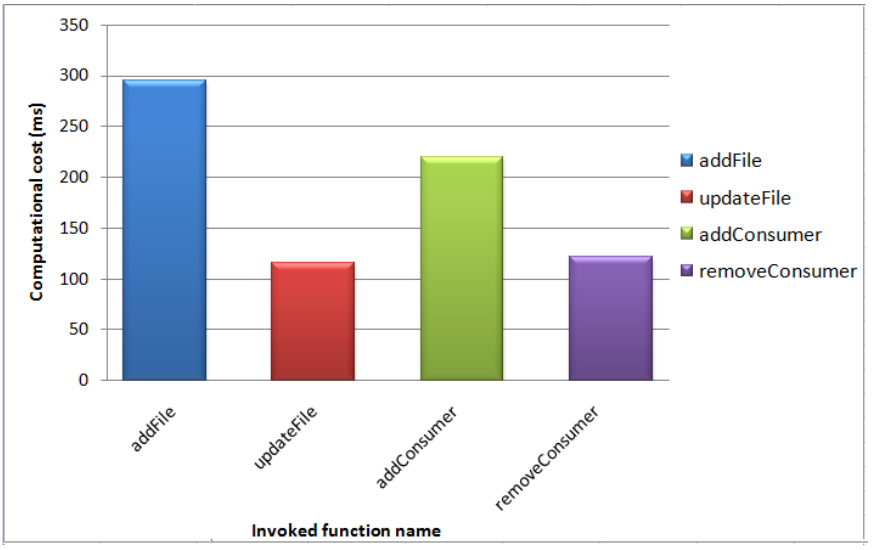


Figure 5.11: Average processing time with different invoked functions

We perform add file operation add by random file contents for 100 repetitions. We measure the required time to compute the file content's hash and invoke the **addFile** function by making several tests while increasing the file size from 1KB to 2MB. Figure 5.12 shows the average processing time of both the **addFile** function invocation and the hash computing time of the file content. We observe that the processing time varies from 257 to 390 ms, which is the higher processing time between the needed time by the three other functions. This can be explained by the necessity to initialize a new state on the smart contract with several information, such as the file name, file path, and the file content's description representing the new file.

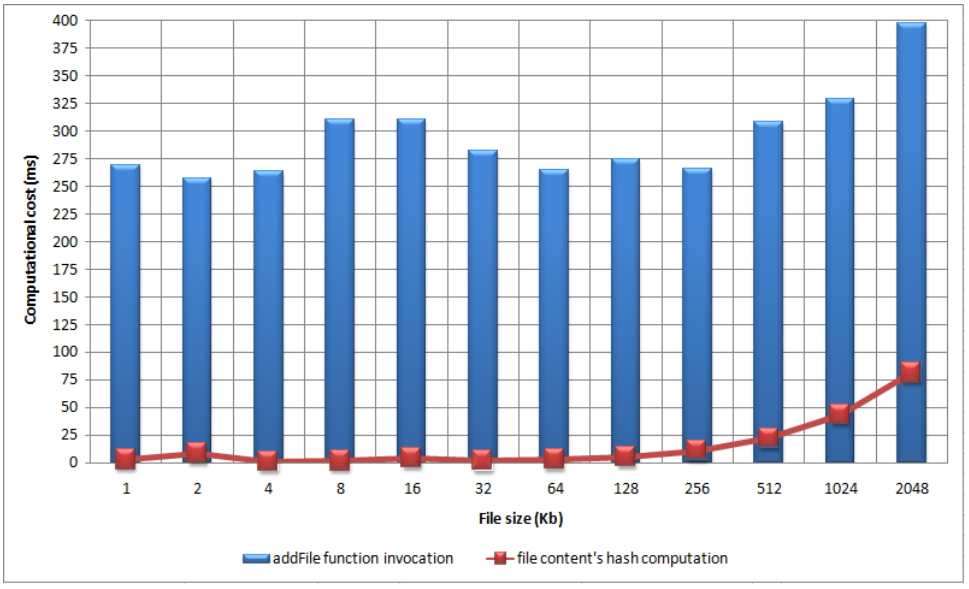


Figure 5.12: Average processing time of **addFile** function with different file size

We perform the same experiment for the **updateFile** function. We measure the required time to compute the file content's hash and store the transaction that invoked the **updateFile** function by making several tests while increasing the file size from 1KB to 2MB. Figure 5.13 shows the average processing time of both the **updateFile** function invocation and the hash computing time of the file content. We observe that the processing time varies from 95 to 180 ms, which is less than the processing time of the three other functions.

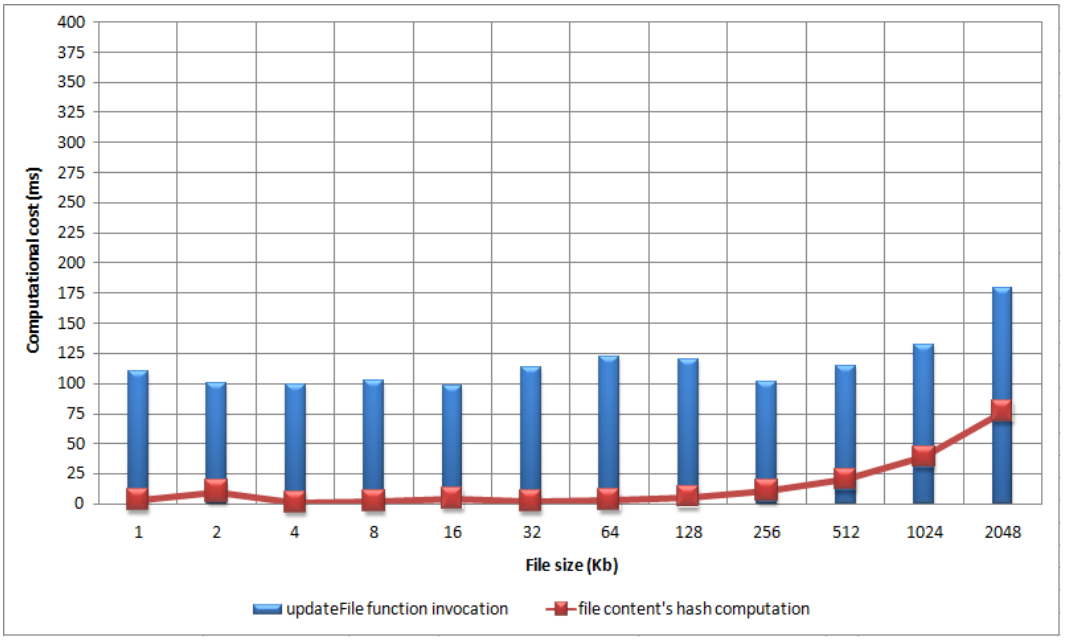


Figure 5.13: Average processing time of **updateFile** function with different file size

5.4.3.2 Cost overhead

In order to evaluate the efficiency of the proposed module, we conduct an experiment to measure the used *gas* quantity by a transaction while invoking one of the IoTDataSharing smart contract's functions, namely **addFile**, **updateFile**, **addConsumer**, and **removeConsumer**.

Figure 5.14 depicts four transaction types according to the invoked function. We observe that the used *gas* by a transaction changed when we change the invoked function. This can be explained by the functions that require more computational resources cost more *gas* than functions that require few computational resources. Moreover, we use in this experiment different file sizes that vary from 1KB to 2MB. We deduce that the used *gas* by the transactions is independent of the file size. Although both **addConsumer** and **removeConsumer** functions are independent of the file size, **addFile** and **updateFile** functions only used the file content's hash whose bit length is fixed and equal to 32 bits. Thus, the file size has no impact on the cost overhead of our proposal. Indeed, this latter can be used in case of files with a huge amount of data without increasing the cost overhead.

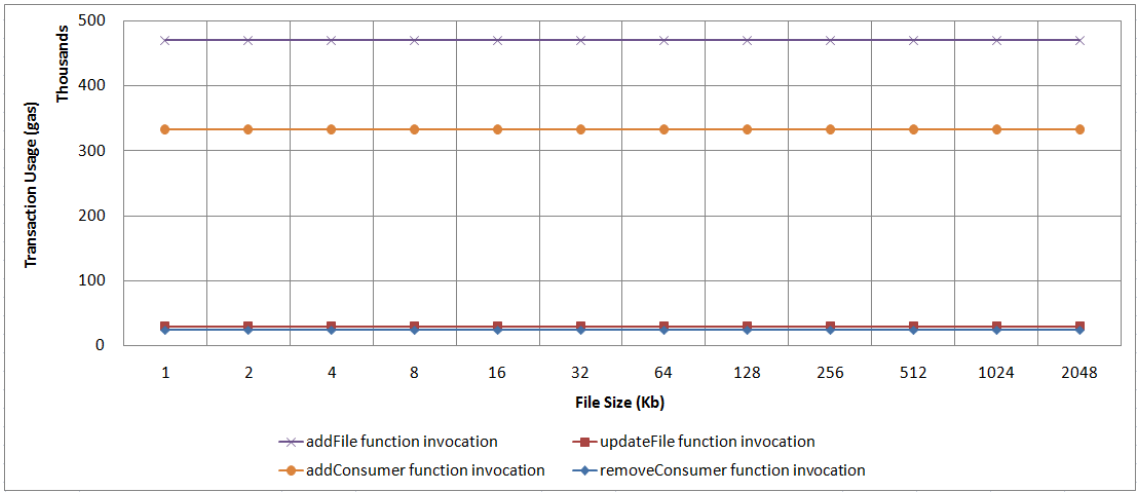


Figure 5.14: Average *gas* usage with different file size

Table 5.1 is the cost overhead of add, update, share, and revoke operations that can be applied on one file using the smart contract functions. The average *gas* usage for a transaction is less than 500000. Currently, 1 *gas* costs about 20 Gwei (i.e., $20 * 10^{-9}$ Ether) and the exchange rate is about 146 EUR for 1 ETHER at the time of writing. Thus, we compute the *gas* cost by multiplying the used *gas* by the *gas* price for transactions that invoke the four smart contract functions. For instance, the *gas* cost of the **addFile** function is equal to the used *gas* multiplied by the *gas* price multiplied by the exchange rate.

$$\begin{aligned} gasCost(EUR) &= usedgas * gasPrice * exchangeRate \\ &= 471959 * 20 * 10^{-9} * 146 \end{aligned} \tag{5.1}$$

Table 5.1: Cost Overhead

Invoked function	Average Usage Cost (*gas*)	Average *Gas* Cost (EUR)
addFile	471959	0,14
updateFile	28946	0,013
addConsumer	332429	0,099
removeConsumer	23456	0,007

We can deduce that the proposed solution can be used in practice and it is not a cost-expensive one.

5.5 Blockchain-based smart grid

In this section, we define a use case for IoT data aggregation, analyze the security and privacy properties, and evaluate the performance of the proposed blockchain-based smart grid.

5.5.1 IoT data aggregation use case

We implement a test system that consists of several nodes, namely 1 key generation node, 1 consumer, 1 aggregator, and 50 smart devices. We assume that each node has a blockchain address.

Let the smart devices as smart meters and the consumer as an energy substation that asks for aggregated smart meter data every 15 minutes for a time duration of 30 days. To implement our use case, we deploy an instance of the **IoTDataSharing** smart contract and interact with it by sending a set of transactions. Figure 5.15 depicts an example of our test system's screenshot. During our experiments, we record the computing time, in milliseconds, of each phase as shown in Figure 5.15. Each phase consists of one or several transactions that invoke the appropriate smart contract's functions in order to read or write on the deployed smart contract.

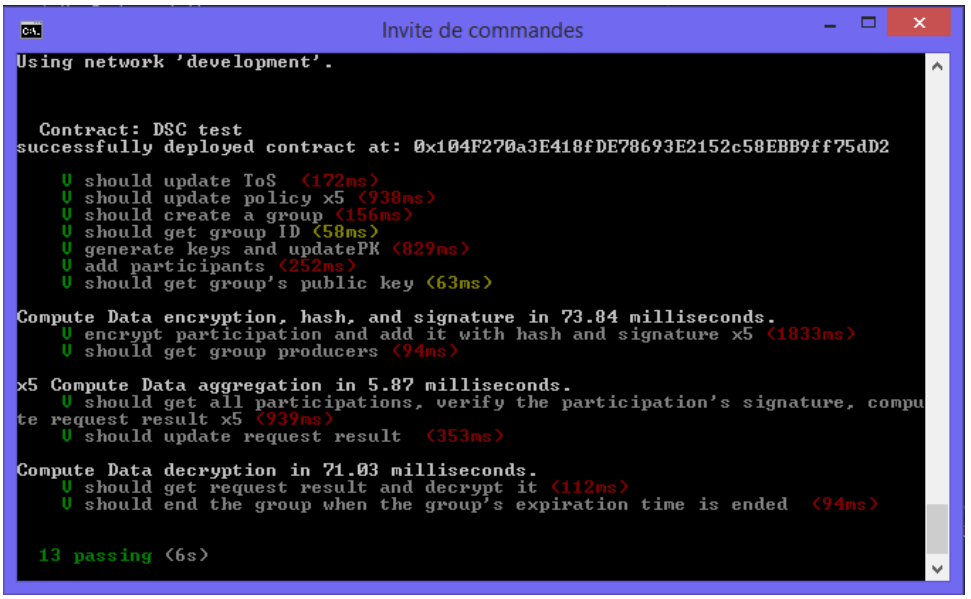


Figure 5.15: Blockchain-based smart grid test system screenshot

First, the energy substation creates a smart contract while indicating the blockchain address of a key generation node. This latter is a JavaScript node that supports the Paillier cryptosystem [Paillier, 1999]. Second, the substation updates its terms of service by invoking the appropriate smart contract function. As mentioned above, the contract events are used to automate the actions taken by the different nodes. Thus, the smart meters update their privacy policies every time the terms of service are updated by invoking the smart contract function, called **updatePrivacyPolicy** using the privacy policy generation JAR file that is proposed based on Algorithm 1, which consists in matching the smart meter owner's privacy preferences and the substation's terms of service in order to generate a common privacy policy about sharing smart meter data. After that, the energy substation creates a group and publishes its request. Once the group is created, the key generation node generates a couple of keys (Pk_{Pai}, Sk_{Pai}), updates the group's public key on the smart contract, and shares the private key with the substation. Based on the privacy policies, the smart contract decides whether or not one smart meter is included in the created group. Once the producers are added to the substation's group, they send period-

ically their produced meter data. Meter data are assumed to be random numbers generated in the range of [0, 4] kilowatt-hour (kWh). Then, the aggregator retrieves all the produced data, checks the integrity and the sender's identity, aggregates them, and updates the request result by invoking the appropriate smart contract function. Once updated, the substation retrieves the request result and decrypts it using the private key Sk_{Pai}. When the retention duration ends, the substation's group is automatically ended by invoking the **endGroup** function.

5.5.2 Security and privacy analysis

After detailing the IoT data aggregation test system, we highlight and analyze in this section both the security and privacy properties.

5.5.2.1 Anonymity and pseudonymity

Hiding the connection between each group's member and its owner's real identity is one of the challenges addressed by the proposed smart grid system.

Anonymity and pseudonymity are a common solution to disguise the user's identity in the blockchain network. However, the connection between the real identity and the pseudonym may be disclosed by matching the individuals' profiles with their behaviors in a particular period of time [Guan et al., 2018]. The proposed smart grid system overcomes this issue by giving each group member the possibility to create and submit the IoT data under different pseudonyms.

5.5.2.2 Data confidentiality, integrity, and sender's identity

The proposed smart grid system can guarantee three security properties, namely the data confidentiality, data integrity, and sender's identity.

For the data confidentiality, only receiver with corresponding private key can recover the encrypted message. Therefore, an adversary eavesdrops the encrypted message, it cannot know the plaintext.

For the data integrity and the sender's identity, the smart contract enables verifying the data's integrity by comparing the received data hash and the computed hash. Moreover, any illegal smart devices can be detected by comparing the sender's blockchain address and the recovered identity using the data hash and its associated digital signature.

To sum up, our smart grid system can ensure that each received message is from the claimed sender, can only be recovered by the intended receiver, and has not been altered during the transmission process.

5.5.2.3 End-to-end privacy-preserving solution

By enabling computation over encrypted IoT data, the result can be computed without revealing the raw IoT data to a consumer or a data aggregator. In this way, the need to trust the consumer or the data aggregator is eliminated during the collection, transmission, storage, and processing phases.

In the data collection, the smart device's data are encrypted using a public key, denoted as Pk_{Pai} of the Paillier cryptosystem [Paillier, 1999]. The used public key is shared by all the group members while no one of the smart devices has the corresponding private key, denoted as Sk_{Pai} to recover the others' ciphertexts. Moreover, each smart device encrypts its generated ciphertext using the chosen aggregator's public key, denoted as Pk_{Agg} before sending its data. Therefore, the plaintext cannot be known by an adversary that dœs not have both the aggregator's private key, denoted as Sk_{Agg} and the private key Sk_{Pai} even if it eavesdrops the ciphertext during the transmission phase. Although the consumer has the private key Sk_{Pai}, it cannot recover the plaintext because it dœs not have the aggregator's private key Sk_{Agg} to decrypt the message.

Moreover, in the data aggregation process, the chosen aggregator just computes the result over the received encrypted data without recovering the individual data of each smart device. Thus, even if the aggregator is compromised, it cannot decrypt the ciphertexts because it dœs not have the appropriate Paillier cryptosystem's private key Sk_{Pai}.

Lastly, when the consumer receives the computed result from the aggregator, it uses the private key Sk_{Pai} in order to recover the final result of the aggregation process. Even if an adversary hacks into the consumer, only the sum of the aggregated data is exposed while smart device's individual data are not disclosed.

To sum up, the proposed smart grid system can ensure the smart device's data privacy during the whole data lifecycle, namely the collection, transmission, storage, and processing phases.

5.5.3 Performance Evaluation

In this section, our proposed smart grid system's performance is evaluated in terms of computation complexity and communication overhead.

5.5.3.1 Computation complexity

We look into the computation complexity in the data processing, which contains three phases, namely data encryption, data aggregation, and data decryption. When the producer wants to send the IoT data, it encrypts the input by the Paillier encryption function, which needs two exponentiation operations in Z_N^* and one multiplication operation. Besides, the data producer performs one hash operation in order to generate a digital signature that enables the verification of both the sender's identity and the data's integrity. When the data aggregator receives all the encrypted data from k producers, it verifies the validity of each sender's identity and data's integrity by performing k hash operations, then it computes the final result by multiplying all the received ciphertexts, which needs $k+1$ multiplication operations, then it executes one hash function before sending the result to the consumer. This latter executes one hash function to verify the received data, then decrypts the ciphertext with the Paillier decryption function, which needs one multiplication operation and one exponentiation operation in Z_N^* in order to recover the plaintext.

Table 5.2 summarizes the computation complexity of the three smart grid system's entities, namely producer, aggregator, and consumer. For simplicity, the exponentiation operation is

denoted as C_e, the multiplication operation is denoted as C_m, and the hash operation is denoted as C_h.

Table 5.2: Computation complexity of IoT data aggregation test system

Entity name	Operations	Computation Complexity
Producer	Data encryption Digital signature generation	$C_m + 2C_e + C_h$
Aggregator	Sender's identity and Data integrity verification Data Aggregation Digital signature generation	$(k+1) * (C_h + C_m)$
Consumer	Sender's identity and Data integrity verification Data decryption	$C_h + C_m + C_e$

In order to measure the performance of our solution, we conduct some experiments to deduce the appropriate number of group members that preserves each member's privacy with a less computational cost. To this end, we evaluate whether the computing time of data aggregation is acceptable by making several tests using a different number of group members that increases from 5 to 50. Hence, we perform a first experiment to measure the required time to check and compute the aggregated data result by an aggregator and a second experiment to measure the required time to decrypt the aggregated data result by a consumer.

Figure 5.16 shows the computational cost of the data aggregation and decryption cases. The computational cost varies from 50 to 440 ms. We observe that data aggregation's computational cost increases with the number of the group members, going from 51 to 440 ms. The cost increases linearly to reach 200 ms at 20 members, keeps the same level until 25 members, and then increases linearly again. This lets us conclude that the appropriate number of group members within a reasonable computational cost is between 20 and 25 members.
Nevertheless, the data decryption computational cost is independent of the number of the group members because all the group members' data are already aggregated by the aggregator. Hence, the consumer receives one ciphertext that represents the sum of all the encrypted group's data.

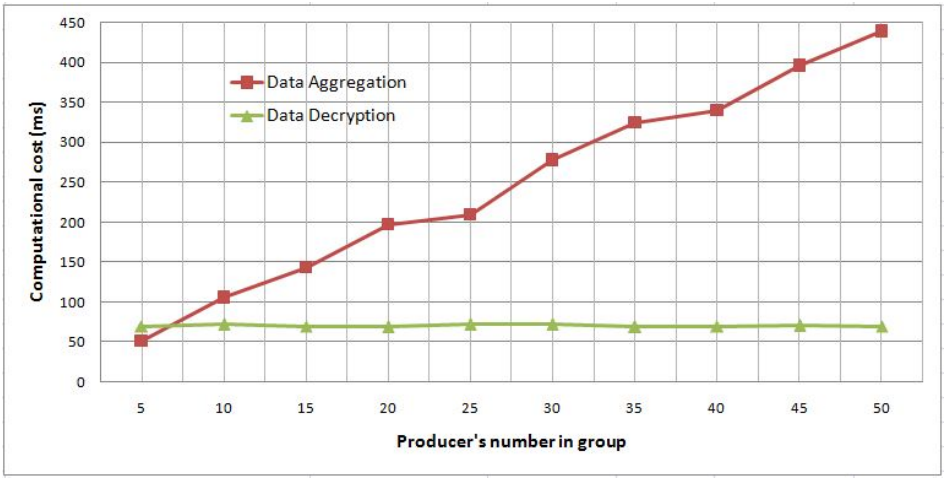


Figure 5.16: Computational cost of data aggregation and data decryption

In order to understand the data aggregation behavior, we split the data aggregation phase

into two parts, namely the smart contract interaction and the data additive homomorphism and conduct a new experiment to measure the required time for each part. As shown in Figure 5.17, the computational cost of the data additive homomorphism part varies only from 7 to 30 ms, as well as the computational cost of the smart contract interaction part varies from 45 to 412 ms, which explains the high data aggregation computational cost in our first experiment depicted in Figure 5.16.

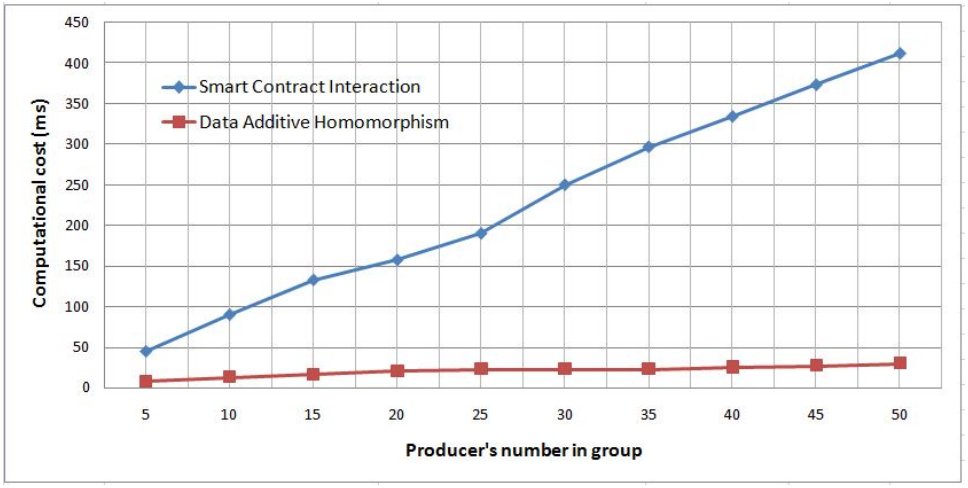


Figure 5.17: Data aggregation's computational cost details

It is worth noting that using a distributed system for storing and accessing data instead of storing all the group members' data on the blockchain would be more appropriate to overcome the computational cost. Thus, the InterPlanetary File System (IPFS) [Benet, 2014] can be used to reduce both the smart contract interaction and the storage costs. IPFS is built on the top of both BitTorrent protocol [Legout et al., 2005] and the Kademlia DHT [Maymounkov and Mazieres, 2002], which are well-known protocols for their ability to scale to a large number of nodes.

5.5.3.2 Communication overhead

The proposed smart grid system enables aggregating raw IoT data from several smart devices into one ciphertext based on the Paillier cryptosystem [Paillier, 1999]. In order to evaluate the efficiency of the proposed solution, we conduct an experiment to measure the communication overhead from the producers to an aggregator, as well as from an aggregator to a consumer. In the proposed smart grid system, the ciphertext's form is $C = g^M r^N \mod N^2$ with the bit length of N is $|N| = 1024$. For the communication from k smart devises to an aggregator, the communication overhead is $k \times 2048$ bits because each smart device encrypts its data to one ciphertext, whose bit length is equal to N^2, i.e., 2048 bits. For the communication from an aggregator to a consumer, the overhead is independent of the smart device number because the data are aggregated on the aggregator before reaching the consumer. Hence, the communication overhead from an aggregator to a consumer is only 2048 bits. Figure 5.18 depicts the communication overhead of the proposed smart grid system.

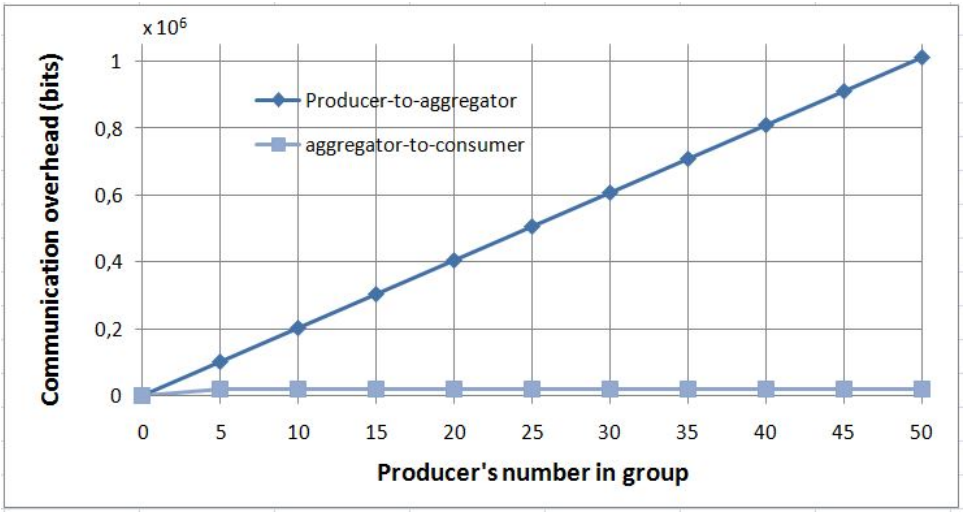


Figure 5.18: Communication overhead

We deduce that the data producers do not need any additional computational capabilities to communicate with the aggregator since each producer sends one data item independently of the number of the group members. However, the aggregator needs more memory and storage capabilities when the group includes more members. To improve our solution, we intend to propose an aggregator selector algorithm that chooses the appropriate group's aggregator according to both the aggregator's computational capabilities and the number of the group's members.

After evaluating our three proposed privacy-preserving test systems, we analyze in the following section the obtained results.

5.6 Comparative performance analysis

In this section, we introduce a comparative performance analysis by comparing our proposed test systems with the existing privacy-preserving approaches in the IoT domain.

Concerning the proposed smart home test system, we compare it with the access control system proposed in [Zhang et al., 2018]. To this end, we implement our smart home scenario to show what benefits it can provide comparing to [Zhang et al., 2018]. This latter is chosen because it is one of the latest approaches offering privacy-preserving access guarantees to the data owner. Besides, it is the more close to our proposal and the authors gave enough details about their implementation. Indeed, we re-implement the work of Zhang et al. [Zhang et al., 2018] according to the methodology described by the authors and the provided source code. Then, we configure it with the same values considered by the authors.

In [Zhang et al., 2018], each couple (subject, object) shared an AccessControlMethod smart contract. The authors defined sending access requests by one subject to the same object too frequently in a short period of time as a misbehavior. Thus, three parameters are defined to characterize this misbehavior, namely *minInterval*, which is the minimum time interval between

two successive requests, *NoFR*, which is the number of frequent requests, and *threshold*, which is the maximum frequent request number in a minimum time interval. In case of a misbehavior detection, the subject is blocked for the duration of time, called *penalty* that is computed by the Judge smart contract using the following function:

$$penalty = (base)^{\ell / interval} \tag{5.2}$$

where ℓ is the number of misbehavior that the subject has exhibited, *base* and *interval* are parameters that determine how the penalty changes with ℓ.

In our case, each object that can be the IoT gateway owned by the data owner defines a BehaviorControl smart contract to manage several subjects (i.e., data consumers). Moreover, we define three misbehavior types, such that (i) sending requests to invoke unauthorized action on one device output, (ii) sending requests during the penalty duration time, and (iii) sending multiple requests in a short period of time. Another difference between our smart home system and [Zhang et al., 2018] is that the BehaviorControl smart contract maintains a history of the previous queries of each device output, thus it can detect the misbehavior of receiving multiple requests from multiple subjects to the same device output in a short period of time. In this case, the device output can be momentary blocked to be protected from this attack. In case of a misbehavior detection, the subject (or the device output) is blocked for the computed duration of time, called *penalty* using the following function:

$$penalty = length / frequencyThreshold \tag{5.3}$$

where *length* is the number of misbehavior that the subject has exhibited and $frequencyThreshold$ is the maximum request number in a short time period.

After detailing the benefits provided by our smart home system comparing to the system proposed in [Zhang et al., 2018], we implement both systems in order to compare their computational cost in terms of CPU usage. Thus, we reuse the two provided JavaScripts in [Zhang et al., 2018] and create two JavaScripts (one at the subject and the other at the object) using the web3.js to interact with the PrivacyPermissionSetting and BehaviorControl smart contracts.

In order to compare the computational cost between the AccessControlMethod and BehaviorControl smart contracts, we evaluate the accessControl smart contract function and our defined smart contract function verifyPermission. Both take as input access request information, namely subject, action, and time, then return the access result retrieved from the appropriate events. The obtained results shows that the proposed smart home system's CPU usage is lower than the CPU usage of the system proposed in [Zhang et al., 2018]. More details about the access results displayed by the Javascripts at the subject side are provided in the appendix (see Appendix B).

Concerning the proposed healthcare test system, we compare its estimated computational costs with Ancile [Dagher et al., 2018] using the *gas* as a unit of measure when comparing on-blockchain operations. Ancile [Dagher et al., 2018] provided a complete process of *adding*

a new record without any computational costs. Thus, we compare the estimated performance differences between this process' steps and our defined **addFile** function invocation's steps.

In Ancile [Dagher et al., 2018], the steps to add a new record for a patient involved:

1- The provider's Database Manager generated a query link to a free location in memory, hashed the link and the record, then sent the link and record to the Cipher Manager.
2- The Cipher Manager generated a symmetric key and encrypted the new record and link, then encrypted the symmetric key with the public keys of the provider, patient, and proxy set.
3- The Database Manager stored the record in the EHR Database.
4- The provider node sent the patient's ID to their Service History Contract.
5- The address of the associated Ownership Contract is returned.
6- The provider node sent the record name, query link hash, record hash, and encrypted symmetric keys.
7- The Ownership Contract created a new Permission Contract for the record and sent the encrypted symmetric keys to the Permission Contract.
8- The new Permission Contract auto-created the provider, patient, and Re-encryption Contract permissions.
9- The Permission Contract sent its address to the Ownership Contract.
10- The record information is added to the Ownership Contract's local memory.
11- The encrypted query link is sent to the patient over HTTPS.
12- The patient node stored the query link in its Cipher Manager.

Steps 1, 2, 3, 11, and 12 are off-blockchain operations, while steps 4, 5, 6, 7, 8, 9, and 10 are on-blockchain operations. The off-blockchain operations involved generating query link, haching it, encrypting it, and database storage/retrieval. The performance cost can be very low depending on the off-blockchain module's implementation. In our case, depending on the off-blockchain module's implementation, the performance cost is lower than Ancile's cost, since the addFile function invocation involves storing the new data, generating a pointer hash of the file location, and encrypting it using the consumer's public key. Concerning the on-blockchain operations, Ancile included several types of operations, such that retrieving and storing data values in smart contracts, sending internal transactions to link the different contracts, and spawning new contracts using other contracts. The *gas* depended on the size of the data values being stored and passed through transactions. In our case, the *gas* is the same in all the transactions since the sent data is a computed hash whose data bit length is fixed.

The process of *adding a new record* proposed by Ancile [Dagher et al., 2018] had more steps compared to our defined addFile function invocation process. Thus, the proposed healthcare test system is lower performance and *gas* costs. Moreover, Ancile allowed placing small records on the blockchain in order to reduce the need for electronic health records databases. However, this feature can violate the data owner's right to be forgotten, since data cannot be deleted once stored on the blockchain. In our case, we rely on off-blockchain decentralized storage in order to address the centralized database management while ensuring the data erasure.

Concerning the proposed smart grid test system, we compare it with a set of aggregation-based privacy-preserving approaches in the IoT domain in Table 5.3. Nine axes are used simultaneously to qualify the aggregation-based approaches, namely the IoT data aggregation,

homomorphic encryption technology support, blockchain technology support, confidentiality check, integrity check, authentication check, anonymity, pseudonymity, and application domain.

Table 5.3: Comparative study between aggregation-based approaches and the proposed smart grid test system

				Security			**Privacy**		
	Data aggregation	**Homomorphic encryption**	**Blockchain**	**Confidentiality**	**Integrity**	**Authentication**	**Anonymity**	**Pseudonymity**	**Application domain**
[González-Manzano et al., 2016]	✓	✓	-	-	✓	✓	✓	-	*Not specified*
[Lu et al., 2017]	✓	✓	-	-	-	✓	-	-	*Not specified*
[Abdallah and Shen, 2018]	✓	✓	-	✓	✓	✓	-	-	Smart Grid
[Liu et al., 2018]	✓	✓	-	✓	✓	✓	-	-	Smart Grid
[Tonyali et al., 2018]	✓	✓	-	-	✓	✓	-	-	Smart Grid
[Guan et al., 2019]	✓	✓	-	-	✓	✓	✓	✓	*Not specified*
[Guan et al., 2018]	✓	-	✓	-	✓	✓	✓	✓	Smart Grid
[Wang et al., 2018]	✓	✓	✓	-	✓	✓	-	-	Smart Grid
Proposed smart grid test system	✓	✓	✓	✓	✓	✓	✓	✓	Smart Grid

Different from the existing proposals (except [Wang et al., 2018]), our proposed smart grid test system combines both the homomorphic encryption and the blockchain technologies in order to preserve IoT data privacy. Thus, our proposal can ensure both security and privacy properties. The difference from our scheme and the two blockchain-based proposals [Guan et al., 2018] [Wang et al., 2018] is the used blockchain platform. In fact, Merkle Tree blockchain system is used in [Guan et al., 2018], hierarchical blockchain system is adopted in [Wang et al., 2018], while the proposed scheme in this paper is based on the Ethereum blockchain because it supports the smart contract use. The reason behind the smart contract use is to (i) enforce a common agreement between several untrusted parties without the involvement of a trusted third party, (ii) organize smart devices into groups according to their owners' privacy choices, and (iii) prevent any identity fraud attempts concerning the smart devices, the aggregator, and the key generation authority.

We compare the overall probability of eavesdropping on private individual data in the proposed smart grid test system and the existing systems considering both a standard centralized system, where the substation received all the smart meters' data to aggregate them and the distributed system proposed by [Wang et al., 2018], where the substation received only an aggregated result of all the smart meters' data. We reuse the same conditions and variables provided in [Wang et al., 2018] to perform this comparison. Thus, we denote success probabilities of manipulating data in a substation and obtaining its private key as γ and $\overline{\gamma}$, respectively. Similarly, we denote variables for both gateway and meter as detailed below in Table 5.4:

Table 5.4: Used variables in the comparison

	Substation	**Gateway**	**Meter**
Hacking into/ Manipulating data in	γ	β	α
Gaining the private key of	$\overline{\gamma}$	$\overline{\beta}$	$\overline{\alpha}$

Table 5.5 shows the success probability of attackers to eavesdrop individual data by each data type considering three system types, namely centralized system, distributed system [Wang et al., 2018], and our proposed smart grid system. We do not consider eavesdropping when it only requires stealing the private key of the data sender. In the following table, we denote unconsidered eavesdropping probability as N/A.

Table 5.5: Comparison of eavesdropping probability of different system components

		Centralized system	**Distributed system**	**Proposed system**
METER	data type	N/A	individual(ciphertext)	individual(ciphertext)
	eavesdropping probability	-	$(\overline{\gamma}\alpha + \overline{\gamma}\mu)/2$	$\overline{\gamma}\alpha$
M-G	data type	individual(ciphertext)	aggregated(ciphertext)	individual(ciphertext)
	eavesdropping probability	N/A	N/A	N/A
Gateway	data type	individual(plaintext)	aggregated(ciphertext)	indi/aggr(ciphertext)
	eavesdropping probability	β	N/A	$\overline{\gamma}\overline{\beta}\beta$
G-S	data type	individual(ciphertext)	aggregated(ciphertext)	aggregated(ciphertext)
	eavesdropping probability	N/A	N/A	N/A
Substation	data type	individual(plaintext)	aggregated(plaintext)	aggregated(plaintext)
	eavesdropping probability	γ	N/A	N/A

For the centralized system, individual data can be eavesdropped by attackers through hacking a gateway or a substation node. Indeed, individual data are not encrypted in these nodes. Thus, the centralized system's eavesdropping probability is equal to $(\beta + \gamma)/2$.

For the distributed system [Wang et al., 2018], individual data can be eavesdropped by attackers through simultaneously gaining the substation's private key and hacking into a meter or a channel between meters. Thus, the distributed system's eavesdropping probability is equal to $(\overline{\gamma}\alpha + \overline{\gamma}\mu)/2$, with μ is the success probability of hacking into a communication channel and gain the sender's private key, a meter in this case, thus $\mu \simeq \overline{\alpha}$, with $0 < \overline{\alpha} < \alpha < 1$.

For the proposed smart grid system, attackers can only eavesdrop individual data through simultaneously (i) gaining the private key of the substation's group and hacking into a meter or (ii) stealing the private keys of both the substation's group and the aggregator and hacking into a gateway. Thus, successful eavesdropping probability in this proposal is $(\overline{\gamma}\alpha + \overline{\gamma}\overline{\beta}\beta)/2$.

Let the substation's defensive capability stronger that the gateway's one; and α is in the range of (0, 1); therefore we deduce $(\overline{\gamma}\alpha + \overline{\gamma}\overline{\beta}\beta)/2 < (\beta + \gamma)/2$, which means that the successful eavesdropping probability in the proposed system is less than the successful eavesdropping probability in the standard centralized system.

$$\left.\begin{array}{l} \overline{\gamma} < \gamma < \overline{\beta} < \beta \\ 0 < \alpha < 1 \end{array}\right\} \Rightarrow \overline{\gamma}\alpha < \gamma \Rightarrow (\overline{\gamma}\alpha + \overline{\gamma}\overline{\beta}\beta)/2 < (\beta + \gamma)/2$$

Let the gateway's defensive capability stronger that the meter's one; hacking into a gateway is the precondition of gaining the node's private key; and hacking into a communication channel between meters is equal to gaining the meter's private key; therefore

$$\left.\begin{array}{l} \overline{\beta} < \beta < \overline{\alpha} \\ 0 < \overline{\beta} < \beta < 1 \\ \mu \simeq \overline{\alpha} \end{array}\right\} \Rightarrow \overline{\beta}\beta < \mu \Rightarrow (\overline{\gamma}\alpha + \overline{\gamma}\overline{\beta}\beta)/2 < (\overline{\gamma}\alpha + \overline{\gamma}\mu)/2$$

Considering all the variables are in the range of (0, 1), then $(\overline{\gamma}\alpha + \overline{\gamma}\overline{\beta}\beta)/2 < (\overline{\gamma}\alpha + \overline{\gamma}\mu)/2$, which means that the successful eavesdropping probability in the proposed system is less than the successful eavesdropping probability in the distributed system proposed in [Wang et al., 2018].

To sum up, the value of a successful eavesdropping probability in the proposed system is less than the considered two systems' values. Thus, the private individual data are better protected in the proposed smart grid test system.

5.7 Summary: privacy design strategies compliance

In this section, we highlight the compliance of our proposals with the privacy design strategies (see section 1.2.2, Chapter 1) according to the three-layered privacy model (see section 1.2.3, Chapter 1).

As aforementioned, the privacy design strategies can be used to evaluate the privacy-preserving approaches in terms of both privacy framework [ISO/IEC29100, 2011] and data protection regulation [GDPR, 2016] compliance. Table 5.6 shows the ensured privacy design strategies by each of the three proposed privacy-preserving test systems. In this table, we denote the centralized architecture as **Cen**, the distributed architecture as **Dis**, and end-to-end data lifecycle as **E2E**.

The smart home test system improves minimizing, hiding, and separating the produced IoT data thanks to the proposed semantic IoT gateway on the user sphere. Besides, the used smart contract improves informing, controlling, and enforcing the data owner's preferences on how the smart devices must behave. Both semantic and blockchain technologies are used to enforce the data owner's control over the owned smart devices.

The healthcare test system improves all the privacy design strategies excepting the aggregate one. In our case, we have only addressed the first use purpose when dealing with the healthcare domain. Thus, data cannot be altered or obfuscated. On the other hand, both semantic and blockchain technologies are used to enforce the data owner's control over the owned data by

Table 5.6: List of the three proposed privacy-preserving test systems

	Architecture	IoT Data Lifecycle	**User Sphere**				**Joint Sphere**		**Recipient Sphere**			
			Minimize	Hide	Separate	Aggregate	Secure Communication	Anonymous Communication	Inform	Control	Enforce	Demonstrate
Smart home test system	Cen+Dis	E2E	✓	✓	✓			✓	✓	✓	✓	
Healthcare test system	Dis	E2E	✓	✓	✓			✓	✓	✓	✓	✓
Smart grid test system	Dis	E2E	✓	✓	✓	✓		✓	✓	✓	✓	✓

allowing consent management, grant and revoke access authorization, and transparency on how data are handled.

The smart grid test system improves all the privacy design strategies. By addressing the second use purpose, data are aggregated on the user sphere rather than disclosed as raw data to the recipient sphere. Moreover, the data owner keeps a level of data ownership and control over the shared data thanks to the blockchain technology use. Both semantic and homomorphic encryption technologies are used to enforce the data owner's anonymity and to eliminate the raw data disclosure issue.

To sum up, the proposed smart contracts improve preserving privacy from an end-to-end view while covering several IoT domains. However, the blockchain use involves both computation and cost overhead. Overall, the additional costs of the privacy-preserving proposals are within the bounds of reasonableness. Given that the test systems are have not yet been optimized.

In summary, this chapter has covered the evaluation and analysis part, which included the creation of three test systems, the results of the experiments, and an analysis of the obtained results. The evaluation was designed to measure a set of measures that was derived from looking at other IoT studies. These measures are essentially the computation, scalability, cost, and communication overheads.

Conclusion and Future Work

Table of Contents

1 **Context and objectives** 151

2 **Proposed contributions** 152

3 **Limitations of this work** 153

4 **Future Work** 154

1 Context and objectives

The Internet of Things (IoT) connects and shares data collected from smart devices in several domains, such as smart home, smart grid, and healthcare. According to Cisco [Cisco, 2016], the number of connected devices is expected to reach 500 Billion by 2030. Five hundred zettabytes of data will be produced by tremendous machines and devices. Usually, these collected data are very sensitive and include metadata, such as location, time, and context. Their analysis allows the collector to deduce personal habits, behaviors and preferences of individuals. Besides, these collected data require the collaboration of several parties to be analyzed. Thus, due to the high level of IoT data sensitivity and lack of trust on the involved parties in the IoT environment, the collected data by different IoT devices should not be shared with each other, without enforcing data owner privacy. In fact, IoT data privacy has become a serious challenge nowadays, especially with the increasing legislation pressure. This context raises the research problems that we address in this thesis.

Thus, our research aims to develop new approaches that focus on four complementary issues that are respectively evaluating and comparing the privacy-preserving approaches in the IoT domain, designing the privacy requirements thanks to an ontology and inference rules, defining a distributed end-to-end system to assist in preserving the IoT data privacy, and evaluating the proposed contributions.

2 Proposed contributions

In this dissertation, we proposed four contributions, each one addressed one of the defined research problems.

A state-of-the-art of the privacy-preserving approaches in the IoT domain.

Based on privacy guidelines, data protection laws, privacy framework, and privacy-related research, we introduced some privacy analysis criteria that are used to compare the privacy preserving approaches in the literature. These criteria included privacy design strategies, a three-layered privacy model, privacy preserving architectures, and privacy preserving mechanisms. Moreover, we surveyed some existing privacy preserving approaches in different domains and we compared them. After that, we provided a review of the literature on privacy preserving approaches proposed in the IoT domain, and we analyzed them using our defined privacy analysis criteria. This analysis showed the lack of an end-to-end solution for privacy in the IoT domain that is compliant with the privacy legislation.

A semantic system to assist in generating privacy requirements.

We proposed LIoPY, a European Legal compliant ontology for supporting preserving IoT PrivacY that described the IoT environment and the privacy requirements. To achieve this, LIoPY imported some concepts from standard ontologies and extended them with new concepts based on privacy legislation. To guarantee a well-built privacy ontology, we followed the MethOntology methodology during LIoPY's process of building. Moreover, we defined a reasoning process whose goal was matching between the data owners' privacy preferences and the data consumers' terms of service. This matching enabled the creation of a common privacy policy that could be applied to preserve the data owner privacy in the IoT environment while handling the shared data. We implemented both LIoPY ontology and the reasoning process, evaluated LIoPY's quality, and analyzed the reasoning process's performance.

A distributed end-to-end system to assist in preserving IoT data privacy.

We introduced PrivBlockchain, an end-to-end privacy-preserving framework for IoT data that aimed at addressing (i) user's control enforcement over the owned smart devices, (ii) privacy requirements and obligation compliance between untrusted parties in the IoT environment, and (iii) individual's privacy rise using group-level IoT data aggregation. To achieve this, PrivBlockchain relied on semantic, blockchain, and homomorphic encryption technologies. Thus, it included three modules, each one addressed one of the aforementioned goals by defining a set of smart contracts. Moreover, we detailed the core processes of each PrivBlockchain's module.

An evaluation through test systems.

We implemented our defined smart contracts and validated them in a blockchain test network. Our evaluation aims at (i) proving that the proposed PrivBlockchain framework is im-

plementable, (ii) providing a performance analysis in terms of processing time, scalability, and cost per transaction, and (iii) deciding whether our solution could be used in practice or it is cost-expensive. To achieve this, we implemented our smart contracts using the Solidity language, deployed them to the Ethereum test network, and created test systems using Truffle development framework. We showed that our PrivBlockchain could handle a large variety of scenarios in different IoT domains.

3 Limitations of this work

Our research does not claim to provide a perfect and indisputable answer to these thesis issues.

Limitations related to the ontology use.

Actually, the IoT data consumers provide their terms of service in form of documents. The idea behind the ontology use is to translate text-based terms of service documents into a computer-processable format. For the first phase of LIoPY ontology definition, we have concentrated on capturing all the key elements of the terms of service documents and building upon them a relevant inference rules in order to generate new knowledge (i.e., generating a new privacy policy). For the LIoPY ontology population phase, only few instances of terms of service are added to the LIoPY ontology. In order to expand LIoPY, text extraction techniques can be used to populate the ontology with more terms of service documents. Besides, it is important to detect any modifications of the original documents in order to update the instances of the LIoPY ontology.

Another limitation related to the algorithm of privacy policy generation through privacy requirement matching (see Algorithm 1) is its output. In our case, a new privacy policy is generated only if all the privacy attributes of the terms of service match all the privacy attributes of the privacy rules of the requested data. Although the obtained result meets the data owner's privacy preferences, the current algorithm is too restrictive. Arranging a negotiation step between the data consumers and the data owners looks an interesting solution for the algorithm's limit. Thus, a new privacy policy can be generated with one of several levels of the data consumer's service according to the frequency/quality of the owner's data. For instance, more the collected data are specific (e.g., value of the age instead of time interval) more the data consumer's service is personalized.

Limitations related to the blockchain technology use.

In our case, the role of a data owner's gateway is to connect the IoT devices to the blockchain network. This gateway node is defined as a device with high memory and storage, which requires a data owner (e.g., a house owner, a hospital, an energy substation) to pay for such a device. The gateway's computing capabilities are related to the number of the managed IoT devices. Thus, the house owner needs a personal computer, but the hospital or the energy substation need a local server. Although the gateway cost differs from one data owner to another, the gateway use will, in each case, increase the cost of using IoT.

4 Future Work

Several areas have still to be explored to address the privacy issue in the IoT domain. In this section, we present two future endeavors.

Optimized Blockchain for IoT.

Applying blockchain technology in the IoT domain is not straightforward due to several challenges including high resource consumption, high memory and storage capabilities, and processing time. Actually, our defined semantic IoT gateway is responsible for a set of smart devices that delegate complicated treatment to the gateway. Thus, the semantic IoT gateway's energy consumption rises when increasing the number of managed smart devices. To address this problem, we intend to employ other blockchain architectures in order to allow resource-constrained devices to participate in the blockchain network and to be more implicated. The impact of this implication on the smart device's battery life needs to be carefully investigated.

Differential privacy.

Blockchain analysis can possibly reveal the frequency of visiting a place or practicing an activity by a specific node. To overcome this problem, our framework enables the use of several addresses for the same IoT resource. Besides, we intend to incorporate the use of differential privacy, a rigorous privacy model that preserves data privacy while maintaining utility in our framework. In fact, by adding some noise to the transactions, we can prevent blockchain analysis.

Furthermore, we intend to incorporate the differential privacy technique in our group-level aggregation to enhance the individual's privacy. The idea behind this is to add noise to the group's members' participation in order to prevent the consumer from inferring extra information when a group's member leaves one group. The impact of the added noise on the data accuracy and the blockchain size needs to be carefully investigated.

Bibliography

Abdallah, A. and Shen, X. S. (2018). A lightweight lattice-based homomorphic privacy-preserving data aggregation scheme for smart grid. *IEEE Transactions on Smart Grid*, 9(1):396–405.

Acar, A., Aksu, H., Uluagac, A. S., and Conti, M. (2018). A survey on homomorphic encryption schemes: theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):79.

Allison, D. S., Kamoun, A., Capretz, M. A., Tazi, S., Drira, K., and ElYamany, H. F. (2016). An ontology driven privacy framework for collaborative working environments. *International Journal of Autonomous and Adaptive Communications Systems*, 9(3-4):243–268.

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM.

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

Bawany, N. Z. and Shaikh, Z. A. (2017). Data privacy ontology for ubiquitous computing. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 8(1):145–149.

Ben-Or, M., Goldwasser, S., and Wigderson, A. (1988). Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM.

Benet, J. (2014). Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*.

Benjamin, C. F., Ke, W., Rui, C., and PHILIP, S. Y. (2010). Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):141–153.

Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., and Taylor, K. (2017). Iot-lite: a lightweight semantic model for the internet of things and its use with dynamic semantics. *Personal and Ubiquitous Computing*, 21(3):475–487.

Bertino, E. (2016). Data security and privacy: Concepts, approaches, and research directions. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, volume 1, pages 400–407. IEEE.

Birman, K., Jelasity, M., Kleinberg, R., and Tremel, E. (2015). Building a secure and privacy-preserving smart grid. *ACM SIGOPS Operating Systems Review*, 49(1):131–136.

Biswas, K. and Muthukkumarasamy, V. (2016). Securing smart cities using blockchain technology. In *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on*, pages 1392–1393. IEEE.

Bogdanov, D., Laur, S., and Willemson, J. (2008). Sharemind: A framework for fast privacy-preserving computations. In *European Symposium on Research in Computer Security*, pages 192–206. Springer.

Burkhart, M., Strasser, M., Many, D., and Dimitropoulos, X. (2010). Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. *Network*, 1(101101).

Buterin, V. et al. (2014). A next-generation smart contract and decentralized application platform. *white paper*.

Castro, M., Liskov, B., et al. (1999). Practical Byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186.

Chen, H., Perich, F., Finin, T., and Joshi, A. (2004). Soupa: Standard ontology for ubiquitous and pervasive applications. In *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 258–267. IEEE.

Cisco (2016). Internet of Things At a Glance. Available online at `https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf?dtid=osscdc000283`. Last accessed: 2019-06-15.

Clarke, R. (2006). What's privacy. In *Australian law reform commission workshop*, volume 28.

Colombo, P. and Ferrari, E. (2015). Privacy Aware Access Control for Big Data: A Research Roadmap. *Big Data Research*, 2(4):145–154.

Cramer, R., Damgård, I., and Maurer, U. (2000). General secure multi-party computation from any linear secret-sharing scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 316–334. Springer.

Cristani, M. and Cuel, R. (2005). A survey on ontology creation methodologies. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 1(2):49–69.

Da Costa, T. M., Martin, H., Rachkidi, E., and Agoulmine, N. (2017). An experiment on deploying a privacy-aware sensing as a service in the sensor-cloud. In *5th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2017)*.

Dagher, G. G., Mohler, J., Milojkovic, M., and Marella, P. B. (2018). Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustainable Cities and Society*, 39:283–297.

Directive 95/46/EC (1995). Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Communities*.

Dorri, A., Kanhere, S. S., and Jurdak, R. (2017a). Towards an optimized blockchain for iot. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pages 173–178. ACM.

Dorri, A., Kanhere, S. S., Jurdak, R., and Gauravaram, P. (2017b). Blockchain for iot security and privacy: The case study of a smart home. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 618–623. IEEE.

Drosatos, G., Tasidou, A., and Efraimidis, P. S. (2017). Privacy-Enhanced Television Audience Measurements. *ACM Trans. Internet Technol.*, 17(1):10:1—-10:29.

Duan, L., Liu, D., Zhang, Y., Chen, S., Liu, R. P., Cheng, B., and Chen, J. (2016). Secure data-centric access control for smart grid services based on publish/subscribe systems. *ACM Transactions on Internet Technology (TOIT)*, 16(4):23.

Dwork, C. (2008). Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472.

EU GDPR (2018). GDPR key changes. Available online at `https://eugdpr.org/the-regulation/`. Last accessed: 2019-06-15.

Fang, W., Wen, X. Z., Zheng, Y., and Zhou, M. (2016). A survey of big data security and privacy preserving. *IETE Technical Review*, pages 1–17.

Fernández-Alemán, J. L., Señor, I. C., Lozoya, P. Á. O., and Toval, A. (2013). Security and privacy in electronic health records: A systematic literature review. *Journal of biomedical informatics*, 46(3):541–562.

Fernández-López, M., Gómez-Pérez, A., and Juristo, N. (1997). Methontology: from ontological art towards ontological engineering. *AAAI Spring Symposium Series*, page 33–40.

Froelicher, D., Egger, P., Sousa, J. S., Raisaro, J. L., Huang, Z., Mouchet, C., Ford, B., and Hubaux, J.-P. (2017). Unlynx: a decentralized system for privacy-conscious data sharing. *Proceedings on Privacy Enhancing Technologies*, 2017(4):232–250.

Funke, S., Daubert, J., Wiesmaier, A., Kikiras, P., and Muehlhaeuser, M. (2015). End-2-end privacy architecture for iot. In *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 705–706. IEEE.

GDPR (2016). Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. *Official Journal of the European Union (OJ)*, 59:1–88.

González-Manzano, L., de Fuentes, J. M., Pastrana, S., Peris-Lopez, P., and Hernández-Encinas, L. (2016). PAgIoT–Privacy-preserving aggregation protocol for Internet of Things. *Journal of Network and Computer Applications*, 71:59–71.

Guan, Z., Si, G., Zhang, X., Wu, L., Guizani, N., Du, X., and Ma, Y. (2018). Privacy-preserving and efficient aggregation based on blockchain for power grid communications in smart communities. *IEEE Communications Magazine*, 56(7):82–88.

Guan, Z., Zhang, Y., Wu, L., Wu, J., Li, J., Ma, Y., and Hu, J. (2019). Appa: An anonymous and privacy preserving data aggregation scheme for fog-enhanced iot. *Journal of Network and Computer Applications*, 125:82–92.

Haller, A., Janowicz, K., Cox, S., Le Phuoc, D., Taylor, K., Lefrançois, M., Atkinson, R., García-Castro, R., Lieberman, J., and Stadler, C. (2017). Semantic Sensor Network Ontology (W3C Recommendation 19 October 2017). `https://www.w3.org/TR/vocab-ssn/`.

Haller, A., Janowicz, K., Cox, S. J., Lefrançois, M., Taylor, K., Le Phuoc, D., Lieberman, J., García-Castro, R., Atkinson, R., and Stadler, C. (2018). The SOSA/SSN ontology: a joint WeC and OGC standard specifying the semantics of sensors observations actuation and sampling. In *Semantic Web*, volume 1, pages 1–19. IOS Press.

Hardjono, T. and Smith, N. (2016). Cloud-based commissioning of constrained devices using permissioned blockchains. In *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security*, pages 29–36. ACM.

Hashemi, S. H., Faghri, F., Rausch, P., and Campbell, R. H. (2016). World of empowered iot users. In *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*, pages 13–24. IEEE.

He, J., Cai, L., Cheng, P., Pan, J., and Shi, L. (2016). Consensus-based privacy-preserving data aggregation. *arXiv preprint arXiv:1609.06381*.

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

Hoepman, J.-H. (2014). Privacy design strategies. In *IFIP International Information Security Conference*, pages 446–459. Springer.

Horridge, M. and Bechhofer, S. (2011). The OWL API: A Java API for OWL Ontologies. *Semantic Web*, 2(1):11–21.

Hosseinzadeh, S., Virtanen, S., Díaz-Rodríguez, N., and Lilius, J. (2016). A semantic security framework and context-aware role-based access control ontology for smart spaces. In *Proceedings of the International Workshop on Semantic Big Data*, page 8. ACM.

Huertas Celdrán, A., Garcia Clemente, F. J., Gil Perez, M., and Martinez Perez, G. (2016). SeCoMan: A Semantic-Aware Policy Framework for Developing Privacy-Preserving and Context-Aware Smart Applications. *IEEE SYSTEMS JOURNAL*, 10(3):1111–1124.

Huertas Celdrán, A., Pérez, M. G., Clemente, F. G., and Pérez, G. M. (2014). Precise: Privacy-aware recommender based on context information for cloud service environments. *IEEE Communications Magazine*, 52(8):90–96.

Iacovazzi, A., D'Alconzo, A., Ricciato, F., and Burkhart, M. (2013). Elementary secure-multiparty computation for massive-scale collaborative network monitoring: A quantitative assessment. *Computer networks*, 57(17):3728–3742.

ISO/IEC29100 (2011). Information technology security techniques privacy framework, ISO/IEC 29100. *ISO JTC 1/SC 27*.

Jayaraman, P. P., Yang, X., Yavari, A., Georgakopoulos, D., and Yi, X. (2017). Privacy preserving Internet of Things: From privacy techniques to a blueprint architecture and efficient implementation. *Future Generation Computer Systems*, pages –.

Kagal, L. and Pato, J. (2010). Preserving privacy based on semantic policy tools. *IEEE Security & Privacy*, 8(4):25–30.

Kravets, R., Tuncay, G. S., and Sundaram, H. (2015). For Your Eyes Only. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, MCS '15, pages 28–35, New York, NY, USA. ACM.

Kshetri, N. (2017). Blockchain's roles in strengthening cybersecurity and protecting privacy. *Telecommunications Policy*, 41(10):1027–1038.

Lai, C., Li, H., Liang, X., Lu, R., Zhang, K., and Shen, X. (2014). Cpal: A conditional privacy-preserving authentication with access linkability for roaming service. *IEEE Internet of Things Journal*, 1(1):46–57.

Lantow, B. (2016). Ontometrics: Putting metrics into use for ontology evaluation. In *KEOD*, pages 186–191.

Legout, A., Urvoy-Keller, G., and Michiardi, P. (2005). Understanding bittorrent: An experimental perspective. .

Li, M., Weng, J., Yang, A., Lu, W., Zhang, Y., Hou, L., Jia-Nan, L., Xiang, Y., and Deng, R. (2018). Crowdbc: A blockchain-based decentralized framework for crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems*.

Li, N., Li, T., and Venkatasubramanian, S. (2007). t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE.

Liang, K., Susilo, W., and Liu, J. K. (2015). Privacy-preserving ciphertext multi-sharing control for big data storage. *IEEE transactions on information forensics and security*, 10(8):1578–1589.

Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., and Njilla, L. (2017). Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 468–477. IEEE Press.

Liu, Y., Guo, W., Fan, C.-I., Chang, L., and Cheng, C. (2018). A practical privacy-preserving data aggregation (3pda) scheme for smart grid. *IEEE Transactions on Industrial Informatics*.

Loukil, F., Ghedira-Guegan, C., Benharkat, A. N., Boukadi, K., and Maamar, Z. (2017). Privacy-aware in the iot applications: A systematic literature review. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 552–569. Springer.

Loukil, F., Ghedira-Guegan, C., Boukadi, K., and Benharkat, A. N. (2018a). Liopy: A legal compliant ontology to preserve privacy for the internet of things. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, pages 701–706. IEEE.

Loukil, F., Ghedira-Guegan, C., Boukadi, K., and Benharkat, A. N. (2018b). Semantic iot gateway: Towards automated generation of privacy-preserving smart contracts in the internet of things. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 207–225. Springer.

Loukil, F., Ghedira-Guegan, C., Boukadi, K., and Benharkat, A. N. (2018c). Towards an end-to-end iot data privacy-preserving framework using blockchain technology. In *International Conference on Web Information Systems Engineering*, pages 68–78. Springer.

Lu, R., Heung, K., Lashkari, A. H., and Ghorbani, A. A. (2017). A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot. *IEEE Access*, 5:3302–3312.

Machanavajjhala, A., Gehrke, J., Kifer, D., and Venkitasubramaniam, M. (2006). l-diversity: Privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24. IEEE.

Maddox, T. (2015). The dark side of wearables: How they're secretly jeopardizing your security and privacy. Available online at `https://www.techrepublic.com/article/the-dark-side-of-wearables-how-theyre-secretly-jeopardizing-your-security-and-privacy/`. Last accessed: 2019-06-15.

Maymounkov, P. and Mazieres, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer.

Mivule, K. (2013). Utilizing noise addition for data privacy, an overview. *arXiv preprint arXiv:1309.3958*.

Nakamoto, S. et al. (2008). Bitcoin: A peer-to-peer electronic cash system.

O'Connor, M., Nyulas, C., Shankar, R., Das, A., and Musen, M. (2008). The SWRLAPI: A Development Environment for Working with SWRL Rules. In *OWLED*.

OECD (1981). *Guidelines on the Protection of Privacy and Transborder Flows of Personal Data*. Organisation for Economic Co-operation and Development.

Onename (2016). Onename project. Available online at `https://www.onename.com/`. Last accessed: 2019-06-15.

Ouaddah, A., Abou Elkalam, A., and Ait Ouahman, A. (2016). Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and Communication Networks*, 9(18):5943–5964.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer.

Pearson, S. (2009). Taking account of privacy when designing cloud computing services. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 44–52. IEEE Computer Society.

Pfitzmann, A. and Köhntopp, M. (2001). Anonymity, unobservability, and pseudonymity—a proposal for terminology. In *Designing privacy enhancing technologies*, pages 1–9. Springer.

Randazzo, F., Croce, D., Tinnirello, I., Barcellona, C., and Merani, M. L. (2015). Experimental evaluation of privacy-preserving aggregation schemes on planetlab. In *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 379–384. IEEE.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

Sacco, O. and Passant, A. (2011). A privacy preference ontology (ppo) for linked data. In *LDOW*. Citeseer.

Seliem, M., Elgazzar, K., and Khalil, K. (2018). Towards privacy preserving iot environments: A survey. *Wireless Communications and Mobile Computing*, 2018.

Sen, A. A. A., Eassa, F. A., Jambi, K., and Yamin, M. (2018). Preserving privacy in internet of things: a survey. *International Journal of Information Technology*, 10(2):189–200.

Shafagh, H., Burkhalter, L., Hithnawi, A., and Duquennoy, S. (2017). Towards blockchain-based auditable storage and sharing of iot data. In *Proceedings of the 2017 on Cloud Computing Security Workshop*, pages 45–50. ACM.

Sharma, M., Chaudhary, A., Mathuria, M., and Chaudhary, S. (2013). A review study on the privacy preserving data mining techniques and approaches. *International Journal of Computer Science and Telecommunications*, 4(9):42–46.

Singla, A., Mudgerikar, A., Papapanagiotou, I., and Yavuz, A. A. (2015). Haa: Hardware-accelerated authentication for internet of things in mission critical vehicular networks. In *MILCOM 2015-2015 IEEE Military Communications Conference*, pages 1298–1304. IEEE.

Sklavos, N. and Koufopavlou, O. (2003). On the hardware implementations of the sha-2 (256, 384, 512) hash functions. In *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS'03.*, volume 5, pages V–V. IEEE.

Solidity (2014). Solidity language. Available online at `https://solidity.readthedocs.io/en/develop/`. Last accessed: 2019-06-15.

Spiekermann, S. and Cranor, L. F. (2009). Engineering privacy. *IEEE Transactions on software engineering*, 35(1):67–82.

Storj (2014). Storj project. Available online at `https://www.storj.io`. Last accessed: 2019-06-15.

Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570.

Tonyali, S., Akkaya, K., Saputro, N., Uluagac, A. S., and Nojoumian, M. (2018). Privacy-preserving protocols for secure and reliable data aggregation in iot-enabled smart metering systems. *Future Generation Computer Systems*, 78:547–557.

Truffle (2016). Truffle: Ethereum development framework. Available online at `https://github.com/trufflesuite/truffle`. Last accessed: 2019-06-15.

Ukil, A., Bandyopadhyay, S., and Pal, A. (2015). Privacy for iot: Involuntary privacy enablement for smart energy systems. In *Communications (ICC), 2015 IEEE International Conference on*, pages 536–541. IEEE.

Wang, S., Hou, Y., Gao, F., and Ma, S. (2016). Ontology-based resource description model for internet of things. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2016 International Conference on*, pages 105–108. IEEE.

Wang, Y., Luo, F., Dong, Z., Tong, Z., and Qiao, Y. (2018). Distributed meter data aggregation framework based on blockchain and homomorphic encryption. *IET Cyber-Physical Systems: Theory & Applications*.

Warren, S. D. and Brandeis, L. D. (1890). The right to privacy. *Harvard law review*, pages 193–220.

Web3 (2017). web3.js - Ethereum JavaScript API. Available online at `https://github.com/ethereum/web3.js/`. Last accessed: 2019-06-15.

Westin, A. F. (1968). Privacy and freedom. *Washington and Lee Law Review*, 25(1):166.

Wong, K.-S. and Kim, M. H. (2014). Towards Self-Awareness Privacy Protection for Internet of Things Data Collection. *JOURNAL OF APPLIED MATHEMATICS*.

Wu, F., Li, X., Xu, L., Kumari, S., Karuppiah, M., and Shen, J. (2017). A lightweight and privacy-preserving mutual authentication scheme for wearable devices assisted by cloud server. *Computers & Electrical Engineering*, pages –.

Yang, J.-J., Li, J.-Q., and Niu, Y. (2015). A hybrid solution for privacy preserving medical data sharing in the cloud environment. *Future Generation Computer Systems*, 43–44:74–86.

Yao, A. C.-C. (1986). How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE.

Youdao, N. (2010). P4p: practical large-scale privacy-preserving distributed computation robust against malicious users. *In Proc USENEX*.

Zhang, Y., Kasahara, S., Shen, Y., Jiang, X., and Wan, J. (2018). Smart contract-based access control for the internet of things. *IEEE Internet of Things Journal*.

Ziegeldorf, J. H., Morchon, O. G., and Wehrle, K. (2014). Privacy in the internet of things: threats and challenges. *Security and Communication Networks*, 7(12):2728–2742.

Zyskind, G., Nathan, O., et al. (2015). Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE.

Appendix A

LIoPY's evaluation using OntoMetrics platform

Listing A.1: XML : LIoPY's evaluation document: OntoMetrics Result

```
<ontometrics-result>
 <ontology id="Optional.of(http://www.owl-ontologies.com/LIoPY.owl)">

 <Basemetrics>
    <Axioms>805</Axioms>
    <Logicalaxiomscount>514</Logicalaxiomscount>
    <Classcount>65</Classcount>
    <Totalclassescount>65</Totalclassescount>
    <Objectpropertycount>52</Objectpropertycount>
    <Totalobjectpropertiescount>52</Totalobjectpropertiescount>
    <Datapropertycount>37</Datapropertycount>
    <Totaldatapropertiescount>37</Totaldatapropertiescount>
    <Propertiescount>89</Propertiescount>
    <Individualcount>56</Individualcount>
    <Totalindividualscount>56</Totalindividualscount>
    <DLexpressivity>ALCOI(D)</DLexpressivity>
 </Basemetrics>

 <Classaxioms>
    <SubClassOfaxiomscount>46</SubClassOfaxiomscount>
    <Equivalentclassesaxiomscount>0</Equivalentclassesaxiomscount>
    <Disjointclassesaxiomscount>0</Disjointclassesaxiomscount>
    <GCICount>0</GCICount>
    <HiddenGCICount>0</HiddenGCICount>
 </Classaxioms>

 <Objectpropertyaxioms>
    <SubObjectPropertyOfaxiomscount>0</SubObjectPropertyOfaxiomscount>
    <Equivalentobjectpropertiesaxiomscount>0</Equivalentobjectpropertiesaxiomscount>
    <Inverseobjectpropertiesaxiomscount>5</Inverseobjectpropertiesaxiomscount>
    <Disjointobjectpropertiesaxiomscount>0</Disjointobjectpropertiesaxiomscount>
    <Functionalobjectpropertiesaxiomscount>0</Functionalobjectpropertiesaxiomscount>
    <Transitiveobjectpropertyaxiomscount>0</Transitiveobjectpropertyaxiomscount>
    <Symmetricobjectpropertyaxiomscount>0</Symmetricobjectpropertyaxiomscount>
```

```
    <Asymmetricobjectpropertyaxiomscount>0</Asymmetricobjectpropertyaxiomscount>
    <Reflexiveobjectpropertyaxiomscount>0</Reflexiveobjectpropertyaxiomscount>
    <Irreflexiveobjectpropertyaxiomscount>0</Irreflexiveobjectpropertyaxiomscount>
    <Objectpropertydomainaxiomscount>52</Objectpropertydomainaxiomscount>
    <Objectpropertyrangeaxiomscount>52</Objectpropertyrangeaxiomscount>
    <SubPropertyChainOfaxiomscount>0</SubPropertyChainOfaxiomscount>
</Objectpropertyaxioms>

<Datapropertyaxioms>
    <SubDataPropertyOfaxiomscount>0</SubDataPropertyOfaxiomscount>
    <Equivalentdatapropertiesaxiomscount>0</Equivalentdatapropertiesaxiomscount>
    <Disjointdatapropertiesaxiomscount>0</Disjointdatapropertiesaxiomscount>
    <Functionaldatapropertyaxiomscount>0</Functionaldatapropertyaxiomscount>
    <Datapropertydomainaxiomscount>37</Datapropertydomainaxiomscount>
    <DataPropertyrangeaxiomscount>37</DataPropertyrangeaxiomscount>
</Datapropertyaxioms>

<Individualaxioms>
    <Classassertionaxiomscount>56</Classassertionaxiomscount>
    <Objectpropertyassertionaxiomscount>221</Objectpropertyassertionaxiomscount>
    <Datapropertyassertionaxiomscount>5</Datapropertyassertionaxiomscount>
    <Negativedatapropertyassertionaxiomscount>0</Negativedatapropertyassertionaxiomscount>
    <Sameindividualsaxiomscount>0</Sameindividualsaxiomscount>
    <Differentindividualsaxiomscount>3</Differentindividualsaxiomscount>
</Individualaxioms>

<Annotationaxioms>
    <Annotationaxiomscount>0</Annotationaxiomscount>
    <Annotationassertionaxiomscount>135</Annotationassertionaxiomscount>
    <Annotationpropertydomainaxiomscount>0</Annotationpropertydomainaxiomscount>
    <Annotationpropertyrangeaxiomscount>0</Annotationpropertyrangeaxiomscount>
</Annotationaxioms>

<Schemametrics>
    <Attributerichness>0.569231</Attributerichness>
    <Inheritancerichness>0.707692</Inheritancerichness>
    <Relationshiprichness>0.530612</Relationshiprichness>
    <Attributeclassratio>0.0</Attributeclassratio>
    <Equivalenceratio>0.0</Equivalenceratio>
    <Axiomclassratio>12.384615</Axiomclassratio>
    <Inverserelationsratio>0.096154</Inverserelationsratio>
    <Classrelationratio>0.663265</Classrelationratio>
</Schemametrics>

<Knowledgebasemetrics>
    <Averagepopulation>0.861538</Averagepopulation>
    <Classrichness>0.2</Classrichness>
</Knowledgebasemetrics>

<Classmetrics>
    <class iri="http://www.owl-ontologies.com/LIoPY.owl#Data_Category"name="Data_Category">
     <Classconnectivity>35</Classconnectivity>
     <Classfulness>0.0</Classfulness>
     <Classimportance>0.160714</Classimportance>
     <Classinheritancerichness>0.0</Classinheritancerichness>
     <Classreadability>0</Classreadability>
```

```
 <Classrelationshiprichness>0</Classrelationshiprichness>
 <Classchildrencount>0</Classchildrencount>
 <Classinstancescount>9</Classinstancescount>
 <Classpropertiescount>0</Classpropertiescount>
</class>

<class iri="http://www.owl-ontologies.com/LIoPY.owl#Privacy_Attribute"
name="Privacy_Attribute">
 <Classconnectivity>0</Classconnectivity>
 <Classfulness>0.0</Classfulness>
 <Classimportance>0.625</Classimportance>
 <Classinheritancerichness>7.222222</Classinheritancerichness>
 <Classreadability>0</Classreadability>
 <Classrelationshiprichness>0</Classrelationshiprichness>
 <Classchildrencount>9</Classchildrencount>
 <Classinstancescount>35</Classinstancescount>
 <Classpropertiescount>0</Classpropertiescount>
 </class>

<class iri="http://www.owl-ontologies.com/LIoPY.owl#Privacy_Rule"name="Privacy_Rule">
 <Classconnectivity>10</Classconnectivity>
 <Classfulness>0.0</Classfulness>
 <Classimportance>0.071429</Classimportance>
 <Classinheritancerichness>0.0</Classinheritancerichness>
 <Classreadability>0</Classreadability>
 <Classrelationshiprichness>0</Classrelationshiprichness>
 <Classchildrencount>0</Classchildrencount>
 <Classinstancescount>4</Classinstancescount>
 <Classpropertiescount>0</Classpropertiescount>
</class>

<class iri="http://www.owl-ontologies.com/LIoPY.owl#Purpose" name="Purpose">
 <Classconnectivity>80</Classconnectivity>
 <Classfulness>0.0</Classfulness>
 <Classimportance>0.214286</Classimportance>
 <Classinheritancerichness>0.0</Classinheritancerichness>
 <Classreadability>0</Classreadability>
 <Classrelationshiprichness>0</Classrelationshiprichness>
 <Classchildrencount>0</Classchildrencount>
 <Classinstancescount>12</Classinstancescount>
 <Classpropertiescount>0</Classpropertiescount>
</class>

<class iri="http://www.owl-ontologies.com/LIoPY.owl#Retention" name="Retention">
 <Classconnectivity>0</Classconnectivity>
 <Classfulness>0.0</Classfulness>
 <Classimportance>0.053571</Classimportance>
 <Classinheritancerichness>0.0</Classinheritancerichness>
 <Classreadability>0</Classreadability>
 <Classrelationshiprichness>0</Classrelationshiprichness>
 <Classchildrencount>0</Classchildrencount>
 <Classinstancescount>3</Classinstancescount>
 <Classpropertiescount>0</Classpropertiescount>
</class>

<class iri="http://www.owl-ontologies.com/LIoPY.owl#Operation" name="Operation">
```

```
      <Classconnectivity>68</Classconnectivity>
      <Classfulness>0.0</Classfulness>
      <Classimportance>0.160714</Classimportance>
      <Classinheritancerichness>0.0</Classinheritancerichness>
      <Classreadability>0</Classreadability>
      <Classrelationshiprichness>0</Classrelationshiprichness>
      <Classchildrencount>0</Classchildrencount>
      <Classinstancescount>9</Classinstancescount>
      <Classpropertiescount>0</Classpropertiescount>
    </class>

    <class iri="http://www.owl-ontologies.com/LIoPY.owl#Disclosure" name="Disclosure">
      <Classconnectivity>26</Classconnectivity>
      <Classfulness>0.0</Classfulness>
      <Classimportance>0.125</Classimportance>
      <Classinheritancerichness>0.0</Classinheritancerichness>
      <Classreadability>0</Classreadability>
      <Classrelationshiprichness>0</Classrelationshiprichness>
      <Classchildrencount>0</Classchildrencount>
      <Classinstancescount>7</Classinstancescount>
      <Classpropertiescount>0</Classpropertiescount>
    </class>

    <class> ... </class>

  </Classmetrics>

  <Graphmetrics>
    <Absoluterootcardinality>23</Absoluterootcardinality>
    <Absoluteleafcardinality>49</Absoluteleafcardinality>
    <Absolutesiblingcardinality>65</Absolutesiblingcardinality>
    <Absolutedepth>134</Absolutedepth>
    <Averagedepth>2.030303</Averagedepth>
    <Maximaldepth>5</Maximaldepth>
    <Absolutebreadth>66</Absolutebreadth>
    <Averagebreadth>3.882353</Averagebreadth>
    <Maximalbreadth>23</Maximalbreadth>
    <Ratioofleaffanoutness>0.753846</Ratioofleaffanoutness>
    <Ratioofsiblingfanoutness>1.0</Ratioofsiblingfanoutness>
    <Tangledness>0.061538</Tangledness>
    <Totalnumberofpaths>66</Totalnumberofpaths>
    <Averagenumberofpaths>13.2</Averagenumberofpaths>
  </Graphmetrics>

  </ontology>
</ontometrics-result>
```

Appendix B

Smart contract-based misbehavior detection results

We conduct experiments to compute the CPU usage and compare the proposed system with the system proposed by Zhang et al. [Zhang et al., 2018]. Thus, we define the same policy with *minInterval* set to 1000 seconds and *frequencyThreshold* set to one. This policy means that the subject can send only one request during 1000 seconds. Otherwise, a misbehavior is detected and a penalty is computed. Moreover, we reuse the same *base* and *interval* in the Judge smart contract as defined by the authors.

Figure B.1 shows the access results displayed by the Javascript at the subject side. While the access is authorized for the first request, a misbehavior is detected when the subject sent a second request before the end of 1000 seconds. We can see that the average of the detected CPU usage is about 23 percent.

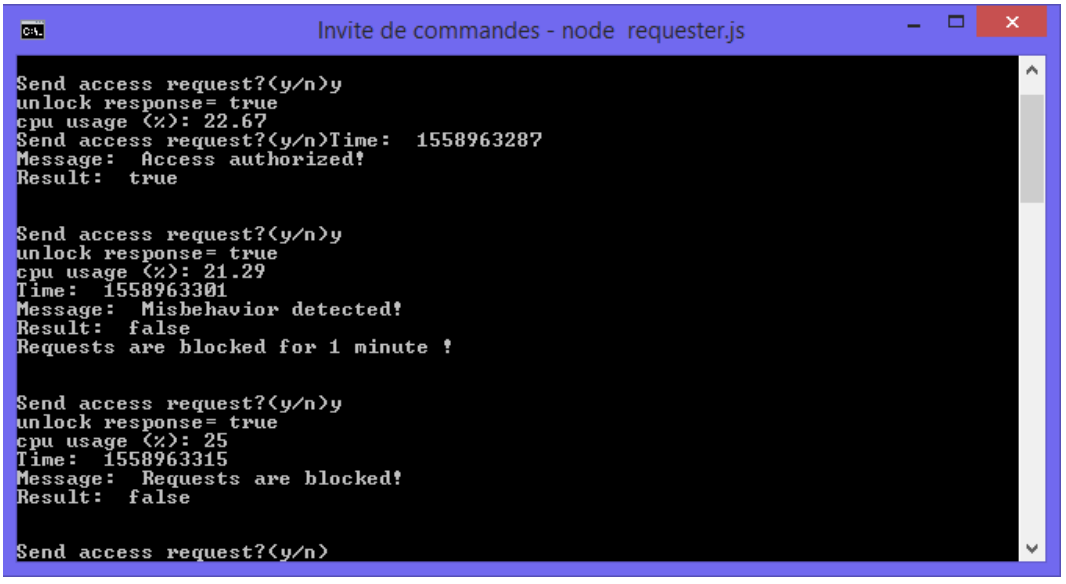


Figure B.1: Results after misbehavior occurring twice by [Zhang et al., 2018] system.

Figure B.2 shows the results, when the subject exhibited the misbehavior for two times. As expected, the subject is blocked for one then two minutes according to the penalty's computation equation 5.3. We can see that the average of the detected CPU usage is about 22,2 percent that is little less than the considered paper [Zhang et al., 2018].

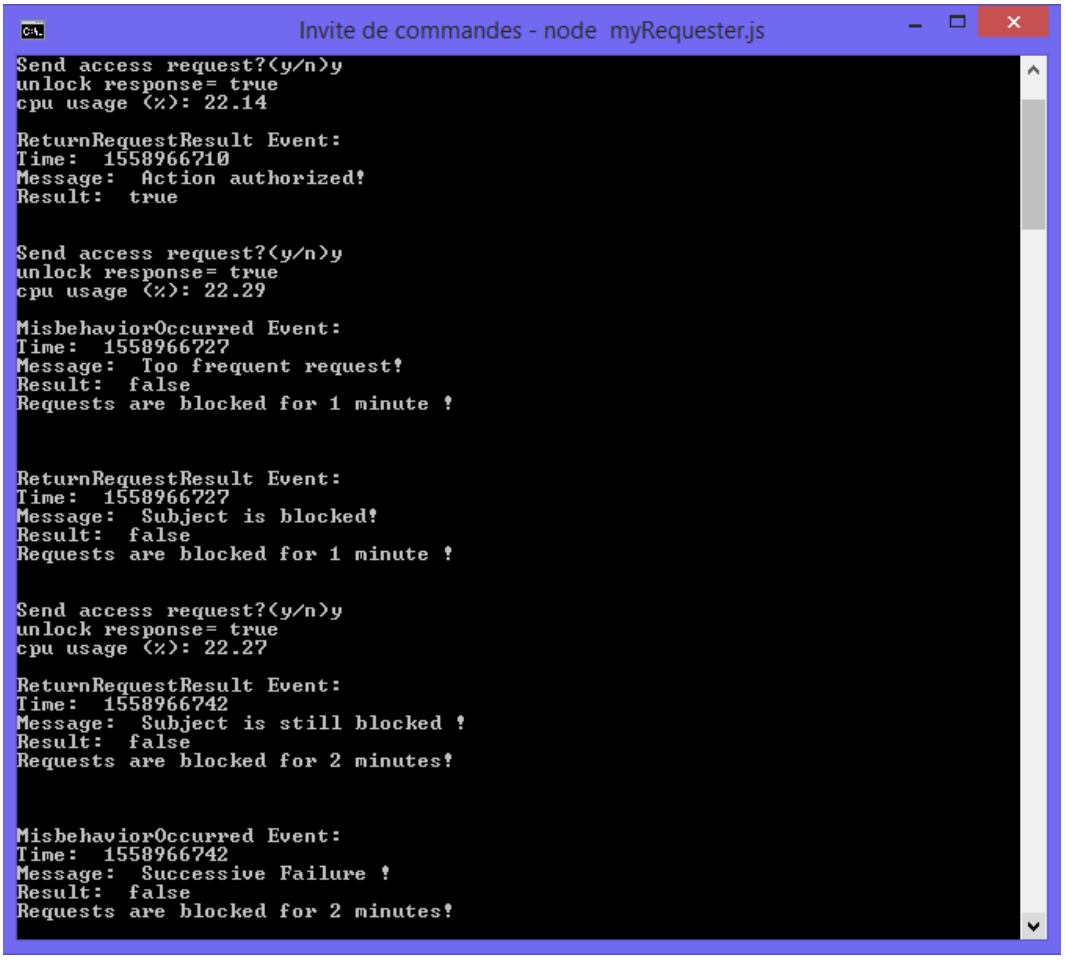


Figure B.2: Results after misbehavior occurring twice by the proposed smart home system.



Title: Towards a new data privacy-based approach for IoT

Abstrct. The Internet of Things (IoT) connects and shares data collected from smart devices in several domains, such as smart home, smart grid, and healthcare. According to Cisco, the number of connected devices is expected to reach 500 Billion by 2030. Five hundred zettabytes of data will be produced by tremendous machines and devices. Usually, these collected data are very sensitive and include metadata, such as location, time, and context. Their analysis allows the collector to deduce personal habits, behaviors and preferences of individuals. Besides, these collected data require the collaboration of several parties to be analyzed. Thus, due to the high level of IoT data sensitivity and lack of trust on the involved parties in the IoT environment, the collected data by different IoT devices should not be shared with each other, without enforcing data owner privacy. In fact, IoT data privacy has become a severe challenge nowadays, especially with the increasing legislation pressure. Our research focused on three complementary issues, mainly (i) the definition of a semantic layer designing the privacy requirements in the IoT domain, (ii) the IoT device monitoring and the enforcement of a privacy policy that matches both the data owner's privacy preferences and the data consumer's terms of service, and (iii) the establishment of an end-to-end privacy-preserving solution for IoT data in a decentralized architecture while eliminating the need to trust any involved IoT parties.

To address these issues, our work contributes to three axes. First, we proposed a new European Legal compliant ontology for supporting preserving IoT PrivacY, called LIoPY that describes the IoT environment and the privacy requirements defined by privacy legislation and standards. Then, we defined a reasoning process whose goal is generating a privacy policy by matching between the data owner's privacy preferences and the data consumer's terms of service. This privacy policy specifies how the data will be handled once shared with a specific data consumer. In order to ensure this privacy policy enforcement, we introduced an IoT data privacy-preserving framework, called PrivBlockchain, in the second research axis. PrivBlockchain is an end-to-end privacy-preserving framework that involves several parties in the IoT environment for preserving IoT data privacy during the phases of collection, transmission, storage, and processing. The proposed framework relied on, on the one hand, the blockchain technology, thus supporting a decentralized architecture while eliminating the need to trust any involved IoT parties and, on the other hand, the smart contracts, thus supporting a machine-readable and self-enforcing privacy policy whose goal is to preserve the privacy during the whole data lifecycle, covering the collection, transmission, storage and processing phases. Finally, in the third axis, we designed and implemented the proposal in order to prove its feasibility and analyze its performances.

Key-Words: privacy, Internet of Things (IoT), ontology and inference, blockchain technology.

Titre : Vers une nouvelle approche respectant la vie privée des données issues des objets connectés

Résumé. Les objets connectés collectent et partagent des données dans différents domaines tels que les maisons intelligentes, les réseaux de distribution d'électricité intelligents et la santé. Selon Cisco, le nombre d'objets connectés devrait atteindre 50 milliards d'ici 2030 avec une quantité de données produites d'environ cinq cents zettaoctets. Toutefois, ces données recueillies sont généralement très riches et comprennent souvent des métadonnées telles que l'emplacement, l'information temporelle et le contexte, rendant ainsi possible de déduire facilement les habitudes personnelles, les comportements et les préférences des individus. De plus, l'analyse de ces données recueillies nécessite la collaboration de plusieurs intervenants. Ainsi, en raison du niveau élevé de la sensibilité des données et du manque de confiance entre les parties impliquées dans un tel réseau, ces données ne doivent pas être partagées, sans que la vie privée du propriétaire des données soit respectée. En effet, la protection de la vie privée des données issues des objets connectés est devenue un défi majeur, en particulier avec la pression croissante de la législation. Nos travaux de recherche sont focalisés sur trois problématiques complémentaires qui sont la problématique de la modélisation des exigences de la protection de la vie privée, la problématique de monitoring les objets connectés et la garantie du respect d'une politique commune qui correspond à la fois aux préférences des propriétaires des données et aux conditions des consommateurs des données, et enfin la problématique de protection de la vie privée durant tout le cycle de vie des données générées par ces objets dans une architecture décentralisée qui élimine le besoin de faire confiance aux parties impliquées dans le réseau d'objets connectés.

Afin de répondre à ces problématiques, nous avons proposé dans un premier lieu une ontologie appelée LIoPY qui modélise la métadonnée ainsi que les contraintes de manipulation des données en adéquation avec les normes et les lois de protection de la vie privée. Puis, pour aligner sémantiquement les exigences en matière de protection de la vie privée des propriétaires des données ainsi que des consommateurs des données, nous avons étendu l'ontologie par des relations sémantiques d'arborescence et des règles sémantiques d'inférence qui génèrent une politique de protection de la vie privée commune. Cette politique décrit comment les données doivent être manipulées une fois partagées avec un consommateur donné. Afin de garantir le respect de cette politique commune, nous avons introduit le framework PrivBlockchain, un framework qui implique toutes les parties intervenantes dans un réseau d'objets connectés dans la protection des données qui en sont issues lors des phases de collecte, du transfert, du stockage jusqu'à la phase de l'utilisation ou bien l'analyse. Le framework proposé repose, d'une part, sur la technologie de la blockchain d'où le support d'une architecture décentralisée, tout en éliminant le besoin de faire confiance aux parties impliquées dans le réseau d'objets connectés et, d'autre part, sur les contrats dits « intelligents » d'où le support d'une politique auto-appliquée et lisible par la machine. Son rôle est de protéger la vie privée lors des phases de collecte, du transfert, du stockage jusqu'à la phase de l'analyse des données issues des objets connectés. Enfin, nous avons validé notre proposition par l'élaboration et l'implantation d'un prototype afin de prouver sa faisabilité et analyser ses performances.

Mots clés : protection de la vie privée, objets connectés, ontologie et inférence, technologie de la blockchain.