Par **Lyes TOUATI**

*Internet of things security: towards a robust interaction of systems of systems*

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC




Soutenue le 21 novembre 2016
**Spécialité** : Technologies de l'Information et des Systèmes :
Unité de recherche Heudyasic (UMR-7253)
D2311

# Sécurité d'internet des objets : vers une interaction robuste des systèmes de systèmes

**Par Lyes Touati**

**Soutenue le 21 novembre 2016**

# Jury Composition

Thèse soutenue le 21 novembre 2016

| | | |
|---|---|---|
| Maryline Laurent | Professeur des Universités | Rapporteur |
| Ahmed Serhrouchni | Professeur des Universités | Rapporteur |
| Lin Chen | Maitre de Conférence | Examinateur |
| Abdelmadjid Bouabdallah | Professeur des Universités | Examinateur |
| Yacine Challal | Maitre de Conférence, HDR | Directeur de Thèse |

# Acknowledgments

I would like to thank GOD for everything. Thank you for your blessings. Thank you for your love. Thank you for my life.

My gratitude to all those who contributed in the achievement of this thesis. Especially my father and all my friends Saif Osamah, Hellaoui Hamed and Azzouzi Oussama.

# Résumé

CETTE thèse traite des problèmes et des défis de sécurité dans l'Internet des Objets (IdO). L'évolution de l'Internet classique vers l'Internet des Objets crée de nombreux challenges dans la manière de sécuriser les communications et soulève des problèmes liés au contraintes de l'internet des objets à savoir : objets à faibles ressources d'énergie et de calculs, hétérogénéité nuisant à l'interopérabilité des objets, taille du réseau de plus en plus grande, ... etc.

En effet, Internet s'est développée d'un réseau d'ordinateurs personnels et de serveurs vers un immense réseau connectant des milliards d'objets intelligents communicants. Ces objets seront intégrés dans des systèmes complexes et utiliseront des capteurs et actionneurs pour observer et interagir avec leur environnement physique.

Les exigences des interactions entre objets communicants en termes de sécurité dépendent du contexte qui évolue dans l'espace et le temps. Par conséquent, la définition de la politique de sécurité doit être adaptative et sensible au contexte.

Un des problèmes auxquels nous nous sommes intéressés est le contrôle d'accès **efficace** à base de cryptographie d'attributs: « Attributes Based Encryption (ABE) ». Les schémas ABE (CP-ABE et KP-ABE) présentent plusieurs atouts pour l'implémentation d'un contrôle d'accès cryptographique. Par contre, ces schémas posent des défis opérationnels à cause de leurs complexités et leur surcoût élevé en termes de temps d'exécution et consommation énergétique. Pour pallier cet inconvénient, nous avons exploité l'hétérogénéité d'environnement Internet des Objets pour proposer des versions collaboratives et distribuées de ces schémas de contrôle d'accès cryptographique. Nos solutions réduisent considérablement le coût en termes d'énergie nécessaire à l'exécution.

Le deuxième inconvénient des schémas ABE est l'inexistence de mécanismes efficaces de gestion de clés. Nous avons proposé des solutions pour le problème de révocation d'attributs dans le schéma CP-ABE, Ces solutions, en plus de leur efficacité, répondent à des exigences de sécurité différentes selon le cas d'applications.

Nous avons proposé également, une solution à base de CP-ABE pour le problème du « grouping proof ». Le « grouping proof » consiste à fournir une preuve sur la coexistence, dans le temps et l'espace, d'un ensemble d'objets. Parmi les applications de notre solution, on peut citer le payement NFC et la sécurisation de l'accès aux locaux sensibles.

# Abstract

IN this thesis, we deal with security chalenges in the Internet of Things. The evolution of the Internet toward a Internet of Things created new challenges relating to the way to secure communications given the new constraints of IoT, namely: resource constrained objects, heterogeneity of network components, the huge size of the network, etc.

Indeed, the Internet evolved from a network of computers and servers toward a huge network connecting billions of smart communicating objects. These objects will be integrated into complex systems and use sensors and actuators to observe and interact with their physical environment.

The security requirements of the interactions between smart objects depend on the context which evolves in time and space. Consequently, the definition of the security policies should be adaptive and context-aware.

In this thesis, we were interested in the problem of access control in IoT relying on Attribute based Encryption (ABE). Indeed, ABE schemes present many advantages in implementing a cryptographic fine-grained access control. However, these schemes raise many implementation challenges because of their complexity and high computation and energy overheads.

To overcome this challenge, we leveraged the heterogeneity of IoT to develop collaborative and distributed versions of ABE schemes. Our solutions reduce remarkably the overhead in terms of energy consumption and computation.

The second limitation of ABE schemes is the absence of efficient attribute/key revocation techniques. We have proposed batch based mechanisms for attribute/key revocation in CP-ABE. We demonstrated the efficiency of the proposed solutions through simulations

Finally, we have propose a CP-ABE based solution for the problem of grouping proof. This problem consist of providing the proof that a set of objects are present simultaneously (same time and same location). The propose solution has many applications such as enforcing the security of NFC based payments and the access to sensitive locations.

# Author Publications List

## International Journal Publications

1. Lyes Touati, Yacine Challal, "Collaborative ABE schemes for Resource-Constrained Devices in Heterogeneous Environments". (Submitted to Computers & Security Journal).

2. Lyes Touati, Yacine Challal, "Efficient and Scalable Key and Attribute Revocation Mechanisms for CP-ABE" (Submitted to Computers & Security Journal).

## International Conference Publications

1. Lyes Touati, "Grouping-Proofs Based Access Control using KP-ABE for IoT Applications". IEEE TrustCom 2017. Sydney, Australia, 2017.

2. Lyes Touati and Yacine Challal, "Proxy-Based Immediate Attribute Revocation Mechanism for CP-ABE in Multicast Group Communications". IEEE TrustCom'2017. Sydney, Australia, 2017.

3. Lyes Touati and Yacine Challal, "Instantaneous Proxy-Based Key Update for CP-ABE," 2016 41st Annual IEEE Conference on Local Computer Networks (LCN 2016), Dubai, United Arab Emirates (UAE), 2016.

4. Lyes Touati, Hamed Hellaoui, and Yacine Challal, "Threshold Yoking/Grouping Proofs based on CP-ABE for IoT Applications", IEEE TrustCom'2016. Tian Jin, China, 2016.

5. Lyes Touati and Yacine Challal, "Collaborative KP-ABE for Cloud-Based Internet of Things Applications," 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, 2016, pp. 1-7.

6. Lyes Touati and Yacine Challal, "Efficient CP-ABE Attribute/Key Management for IoT Applications," in 2015. The 15th IEEE International Conference on Computer and Information Technology (IEEE-CIT'15), Liverpool, UK, Oct. 2015.

7. Lyes Touati and Yacine Challal. 2015. "Poster: Activity-Based Access Control for IoT". In Proceedings of the 1st International Workshop on Experiences with the Design and Implementation of Smart Objects (SmartObjects@MobiCom 2015). ACM, New York, NY, USA, 29-30.

8. Lyes Touati and Yacine Challal, "Batch-Based CP-ABE with Attribute Revocation Mechanism for the Internet of Things," in 2015 International Conference on Computing, Networking and communications, Wireless Networks Symposium (IEEE-ICNC'15 WN), Anaheim, USA, Feb. 2015.

9. Lyes Touati, Yacine Challal and Abdelmadjid Bouabdallah, "C-CP-ABE: Cooperative Ciphertext Policy Attribute-Based Encryption for the Internet of Things," in Advanced Networking Distributed Systems and Applications (INDS), 2014 International Conference on, June 2014.

## Talks and Presentations

1. Lyes Touati, Internet of things Security, Seminar at ICube Laboratory, Strasbourg. April 10th, 2017.

2. Lyes Touati, THreshold Grouping Proofs Based Access Control for IoT, Seminar at CentraleSupelec. University of Paris-Sud. March 24th, 2017.

3. Lyes Touati, Hamed Hellaoui and Yacine Challal, Contrôle d'accès basé sur la "Preuve de Coexistence d'objets" pour l'IoT, Non Themathic days Rescom, Inria Sophia-Antipolis. January 12th, 2017.

4. Lyes Touati and Yacine Challal, Contrôle d'activité des objets dans l'Internet des Objets, Thematic " smart cities " day Rescom, Inria Sophia-Antipolis. January 11th, 2017.

# Glossary

| | |
|---|---|
| 3G | Third Generation |
| 4G | Fourth Generation |
| 6LoWPAN | IPv6 over Low power Personal Area Networks |
| ABE | Attribute Based Encryption |
| CP-ABE | Ciphertext-Policy Attribute-Based Encyrption |
| ETSI | European Telecommunications Standards Institute |
| ICT | Information and Communication Technologies |
| IETF | Internet Engineering Task Force |
| IoT | Internet of Things |
| IPv4 | Internet Protocol version 4 |
| IPv6 | Internet Protocol version 6 |
| KP-ABE | Key-Policy Attribute-Based Encryption |
| LLN | Low Power Lossy Networks |
| M2M | Machine-to-Machine |
| NIC | US National Intelligence Council |
| RFID | Radio Frequency IDentification |
| RPL | IPv6 Routing Protocol for Low Power and Lossy Networks |
| TVoIP | Television Over Internet Protocol |
| VoIP | Voice over IP |
| WiMAX | Worldwide Interoperability for Microwave Access |
| WLAN | Wireless Local Area Network |
| xDSL | Digital Subscriber Line |

# Contents

**Jury Composition** **iii**

**Acknowledgments** **v**

**Résumé** **vii**

**Abstract** **ix**

**Author Publications List** **xi**

**Glossary** **xiii**

**Contents** **xv**

**List of Tables** **xix**

**List of Figures** **xxi**

**1 Introduction** **1**

1.1 Challenges and Objectives . . . . . 1

1.2 Contributions . . . . . 2

1.3 Outline . . . . . 2

**2 Internet of Things : Opportunities and Security Challenges** **5**

2.1 Introduction . . . . . 5

2.2 Internet of Things . . . . . 5

2.2.1 IoT Applications . . . . . 6

2.2.2 IoT Architecture . . . . . 6
2.3 Internet of Things Security . . . . . 7
2.3.1 IoT Vulnerabilities and Threats . . . . . 8
2.3.2 IoT Security and Privacy Challenges . . . . . 9
2.4 Access control in IoT . . . . . 11
2.4.1 Definition . . . . . 11
2.4.2 Classification of Access Control Solutions for IoT . . . . . 12
2.4.3 Discussion . . . . . 14
2.5 Conclusion . . . . . 14

**3 Attribute Based Encryption in IoT** **15**
3.1 Introduction . . . . . 15
3.2 Attribute Based encryption . . . . . 15
3.2.1 Overview . . . . . 15
3.2.2 Preliminaries . . . . . 17
3.2.3 ABE schemes . . . . . 17
3.3 Challenges of implementing ABE in IoT . . . . . 21
3.3.1 Resource limitations . . . . . 21
3.3.2 Key/Attribute Revocation . . . . . 22
3.4 Optimized Attribute based Encryption for resource constrained networks . . . . . 22
3.5 Key/Attribute Revocation in ABE . . . . . 25
3.6 Conclusion . . . . . 27

**4 Collaborative ABE schemes in IoT** **29**
4.1 Introduction . . . . . 29
4.2 Network model and assumptions . . . . . 29
4.2.1 Network model . . . . . 29
4.2.2 Assumptions . . . . . 31
4.3 Collaborative CP-ABE . . . . . 31
4.3.1 Per-phase Collaborative Encryption Primitive . . . . . 31

4.4 Collaborative KP-ABE . . . . . 34
4.4.1 Per-phase Collaborative Encryption Primitive . . . . . 34
4.5 Security Analysis . . . . . 36
4.5.1 Collaborative CP-ABE . . . . . 36
4.5.2 Collaborative KP-ABE . . . . . 36
4.6 Performance Analysis . . . . . 37
4.6.1 Experiments Settings . . . . . 37
4.6.2 Number of Operations . . . . . 37
4.6.3 Execution time cost . . . . . 39
4.6.4 Communication cost . . . . . 41
4.6.5 Energy consumption . . . . . 42
4.7 Applying delegation to the decryption primitives . . . . . 44
4.7.1 Principle . . . . . 45
4.7.2 Delegation for CP-ABE decryption . . . . . 47
4.7.3 Delegation for KP-ABE decryption . . . . . 48
4.8 Conclusion . . . . . 48

**5 Attribute Revocation Mechanisms in ABE** **55**

5.1 Introduction . . . . . 55
5.2 Splitting Time Axis into Time Slots . . . . . 55
5.2.1 Model and Requirements . . . . . 55
5.2.2 Batch Based CP-ABE . . . . . 57
5.2.3 Variable Time Slots Durations . . . . . 62
5.2.4 Performance Analysis . . . . . 66
5.3 Instantaneous Proxy revocation . . . . . 72
5.3.1 Network model . . . . . 72
5.3.2 Assumptions . . . . . 74
5.3.3 Solution 1: Backward and Forward Accessibility . . . . . 75
5.3.4 Solution 2: Backward and Forward Secrecy . . . . . 85
5.4 Conclusion . . . . . 90

**6 Threshold grouping proof in IoT** **93**

6.1 Introduction . . . 93

6.2 Yoking/Grouping Proof Techniques . . . 93

6.3 CP-ABE based Threshold Grouping proof . . . 94

6.3.1 Overview . . . 94

6.3.2 Network Model . . . 95

6.3.3 Security requirements . . . 95

6.3.4 Construction . . . 96

6.3.5 Group dynamics . . . 99

6.3.6 Managing many groups of entities . . . 100

6.3.7 Security Analysis . . . 100

6.3.8 Advantages . . . 101

6.3.9 Performance Analysis . . . 102

6.4 Application cases . . . 104

6.5 Conclusion . . . 105

**7 Conclusions and future directions** **107**

7.1 Conclusions . . . 107

7.2 Open Issues and Future directions . . . 108

**8 Appendix** **111**

8.1 Appendix A: Proof $ANT \approx 1/\mu * \Delta t$ . . . 111

8.2 Appendix B: Proof $AWT \approx \Delta t/2$ . . . 114

8.3 Appendix C: A simple definition of the our hash-function . . . 117

**Bibliography** **119**

# List of Tables


3.1 Comparative table . . . . . 25

3.2 Comparative table . . . . . 27

4.1 Size of the elements in different pairings . . . . . 37

4.2 Computation comparison . . . . . 38

4.3 Computation comparison . . . . . 38

4.4 Comparison between CP-ABE and C-CP-ABE ($p = 5$) . . . . . 49

4.5 Impact of the number of assistant nodes in C-CP-ABE ($|Y_{cpabe}| = 50$) . . . . . 50

4.6 Comparison between KP-ABE and C-KP-ABE ($p = 5$) . . . . . 51

4.7 Impact of the number of assistant nodes in C-KP-ABE ($|\gamma| = 100$) . . . . . 52

4.8 Energy consumption for CP-ABE and C-CP-ABE with respect to the number of assistant nodes $p$ . . . . . 52

4.9 Comparison in terms of Energy consumption . . . . . 52

4.10 Comparison between KP-ABE and C-KP-ABE in terms of Energy consumption . 53

4.11 Comparison in terms of Energy consumption . . . . . 53

5.1 Example . . . . . 59

5.2 Example. . . . . 63

5.3 Proxy data structure . . . . . 77

5.4 Size of elements for different pairings . . . . . 82

5.5 Required storage size for the proxy to store users' table . . . . . 82

5.6 Size of the user's secret key . . . . . 83

5.7 Size of exchanged messages during decryption . . . . . 84

5.8 Proxy data structure . . . . . 86

6.1 Comparison between approaches . . . . . 102

6.2 Performance Analysis: Number of operations . . . . . 102

6.3 Elements Sizes . . . . . 103

6.4 Performance Analysis. Required Storage Capacity . . . . . 104

# List of Figures

2.1 IoT/M2M standardization landscape. Source ETSI presentation at MWC 2011[30] 7

2.2 Classification of Access Control Solutions . . . . . 12

3.1 (a) Example of access tree. (b) KP-ABE example. (c) CP-ABE example . . . . . 16

3.2 Classification of Attribute Revocation solutions for CP-ABE . . . . . 25

4.1 Network model . . . . . 30

4.2 C-CP-ABE Scheme . . . . . 32

4.3 C-KP-ABE Scheme . . . . . 34

4.4 Comparison between CP-ABE and C-CP-ABE in term of execution time ($p = 5$) 39

4.5 Impact of the number of assistant nodes in C-CP-ABE . . . . . 40

4.6 Comparison between KP-ABE and C-KP-ABE in term of execution time ($p = 5$) 40

4.7 Impact of the number of assistant nodes in C-KP-ABE . . . . . 41

4.8 Maximum number authorized of assistant nodes for C-CP-ABE without losing in communication performances . . . . . 43

4.9 Maximum number authorized of assistant nodes for C-KP-ABE without losing in communication performances . . . . . 44

4.10 Energy consumption of CP-ABE and C-CP-ABE with respect to the number of assistant nodes $p$ . . . . . 45

4.11 Comparison in terms of Energy consumption with respect to number of attributes in $Y_{cpabe}$ . . . . . 45

4.12 Comparison in terms of Energy consumption with respect to the number of assistant nodes $p$ . . . . . 46

4.13 Comparison in terms of Energy consumption with respect to number of attributes 46

5.1 Example . . . . . 59
5.2 Example of creating time slots with variable durations . . . . . 63
5.3 Average number of time slots . . . . . 67
5.4 Average number of time slots . . . . . 68
5.5 Average waiting time . . . . . 68
5.6 AWT vs ANT . . . . . 69
5.7 Number of generated secret key parts SKP with respect to $\lambda$ . . . . . 70
5.8 Number of generated secret key parts SKP with respect to $\mu$ . . . . . 71
5.9 Number of generated secret key parts SKP with respect to both $\lambda$ and $\mu$ . . . . . 72
5.10 Comparison with respect to $\lambda$ . . . . . 73
5.11 Comparison with respect to $\mu$ . . . . . 74
5.12 Architecture of the solution . . . . . 75
5.13 Sequence Diagram of different phases of our scheme . . . . . 78
5.14 Required storage size with respect to number of users . . . . . 83
5.15 Size of user's secret key ($n = 50, a = 20$) . . . . . 84

6.1 System configuration phase . . . . . 97
6.2 Recovering secret process . . . . . 99
6.3 Number of operations executed during the decryption process by each entity . . . 103
6.4 Size of secret elements for both the proxy and an entity in both of our approach and original CP-ABE . . . . . 104
6.5 Illustration of the location access control system . . . . . 105

# Chapter 1

# Introduction

The Internet is one of the 20th century's disruptive technologies that have had a deep impact on our daily lives. Like water and electricity, the Internet has become unavoidable in the pursuit of day-to-day activities: e-commerce, e-banking, e-insurance, e-learning, TVoIP, VoIP, e-conferencing, etc.

Recent advances in Information and Communication Technologies (ICT) and Embedded Systems have given rise to a new disruptive technology in the 21st century: The Internet of Things (IoT), that by 2020 [65] will have transformed the Internet from a network of 2.5 billion PCs and smartphones into a global network interconnecting tens of billions of communicating objects. These objects will be integrated into complex systems and use sensors and actuators to observe and interact with their physical environment, and hence allowing interaction among autonomous systems.

## 1.1 Challenges and Objectives

The Internet of Things (IoT) raises important questions and introduces new challenges for the security of systems and processes and the privacy of individuals. Some Internet of Things applications are tightly linked to sensitive infrastructures and strategic services such as the distribution of water and electricity and the surveillance of bridges and buildings. Other applications handle sensitive information about people, such as their location and movements, or their health and purchasing preferences. Confidence in and acceptance of IoT will depend on the protection it provides to people's privacy and the levels of security it guarantees to systems and processes. An urgent prerequisite for securing IoT is the development of efficient security mechanisms for tiny embedded networks with scarce resources. Current developments in wireless sensor and actuator networks, RFID technology, mobile computing and so forth, demonstrate the resource scarcity of the devices and technologies that will be part of IoT. The ubiquitous nature of IoT raises legitimate questions about the privacy of persons, and how to cope with the heterogeneity of user and application requirements in terms of security services. This requires the development of adaptive, context-aware and user-centric security solutions

This thesis aims to develop a new global approach for IoT security that takes into consideration the involvement of smart communicating objects in the control of complex systems and the ubiquitous nature of IoT. Security requirements in System of Systems interactions will depend on

the context that evolves in space and time. Therefore, security policy definition and enforcement should be adaptive and "context-aware". This will allow the development of efficient security solutions for robust interaction of smart objects with persons, the technological ecosystem and control processes, while providing autonomy for objects to safely perceive and act on their environment.

## 1.2 Contributions

After reviewing security threats against sensitive IoT applications, we focused the aim of this thesis on access control services. In particular, we considered attribute based encryption (ABE) [11] [34] as a basis for the development of fine-grained cryptographic access control for IoT. Indeed ABE schemes provide high expressiveness with respect to access policies definition which allows to better consider the context in policy access definition and update. However, ABE schemes suffer from their complexity and heavy overhead. They also do not provide efficient key/attribute revocation mechanisms. The aim of this thesis was then to overcome those challenges to pave the way for efficient fine-grained access control that supports objects and users dynamism and context-aware access policy definition.

The contributions of this thesis are then manifolds:

1. First, we have proposed distributed and collaborative versions of two powerful Attribute-Based Encryption schemes (CP-ABE and KP-ABE). We provided security analysis and proved the safely of our solutions. Moreover, we carried out experiments to evaluate the performances of our solutions and demonstrate their applicability in the constrained environement of IoT.

2. Then, we proposed an efficient batch-based attribute revocation mechanism for CP-ABE. Batch revocation optimizes indeed the overhead due to re-keying and attribute reassignment. It is applicable in the case where revocation schedule is known a priori or when it tolerates some delay before becoming effective. We propose therefore different variants depending on the applications' requirements and revocation schedule assumptions.

3. Finally, we have proposed a CP-ABE based solution for threshold grouping proofs problem. Our solution can provide efficiently the proof of the presence of a threshold of $k$ objects among a group of $N$ objects. This solution is very interesting to strengthen security of IoT applications that provide sensitive services (ex. NFC payment, Local access management) or manage sensitive data.

## 1.3 Outline

This thesis is organized as follows:

- In Chapter 2 we present an overview of Internet of Things security challenges. After discussing security threats we highlight specific security requirement for IoT. Then we focus the discussion on acccess control services given the sensitivity of targeted applications. There, we survey existing access control approaches and solutions and present a classification of existing techniques. Amongst existing techniques we pointed out advantages of attribute based encryption in terms of providing fine-grained access control.

- In Chapter 3, we present background material relating to attribute based encryption and its variants: CP-ABE and KP-ABE. Then, we highlighted challenges that brake implementing those schemes in the context of IoT. Further, we surveyed existing solutions aiming to overcome those challenges. In particular we considered solutions aiming to overcome resource constraints of IoT that prevent smooth implementation of ABE, and solutions aiming to provide efficient key/attribute revocation in spite of the huge size of IoT and dynamism of smart objects and users.

- In Chapter 4, we propose our own solutions for the problem of resource constraints. We leveraged IoT heterogeneity to offload heavy ABE computations such as exponentiation to powerful components of the network. We presented thorough collaborative versions of CP-ABE and KP-ABE as well as experimental results that corroborate the applicability of our solutions the severely constrained environment of IoT.

- In Chapter 5, we present our solutions for key/attribute revocation challenge based on a batch re-keying / attribute re-assignment mode.

- Chapter 6 is devoted to the problem of grouping proof. We present in this chapter a CP-ABE based grouping proofs solution.

- Finally, we conclude this thesis in Chapter 7 and shed some light on open issues and future directions.

# *Chapter 2*

# Internet of Things : Opportunities and Security Challenges

## 2.1 Introduction

Internet of Things (IoT) [9] is an enabling technology for Cyber-Physical Systems or Systems of Systems. Indeed, Internet is evolving from a network of personal computers and servers toward a huge network interconnecting billions of smart communicating objects. These objects will be integrated into complex systems and use sensors and actuators to observe and interact with their physical environment, and hence allowing interaction among autonomous systems. IoT will be involved in various applications ranging from military [69], [58], [58] (military logistics, enemy territories exploration, soldiers monitoring, ...), to e-health [61], [10] (monitoring elder-lies, remote diagnosis, ...), smart cities [60]- [44], smart grid [74], smart vehicles and transportation (traffic jam management), etc. Given the sensitivity of IoT applications, access control becomes a compulsory security service to prevent attacks against those sensitive applications that would have a deep impact and great damages on the systems themselves and their users and customers. Privacy is another issue that requires fine-grained access control to avoid access to private information by third parties.

However, what should be protected from disclosure depends heavily on user's and/or system's context. Moreover, the access policy may evolve following a change in the user's context. What must remain confidential under some circumstances, may be a vital input for a third party for user's safety and security. For instance, a person may deny access to her/his location to preserve her/his privacy. If it comes that the same person falls in an isolated location and needs help, activation to her/his location may be vital. Therefore, access control policy must be adaptive and context-aware.

## 2.2 Internet of Things

The Internet of Things (IoT) is a novel paradigm that is rapidly gaining ground in the scenario of modern wireless telecommunications. The basic idea of this concept is the pervasive presence around us of a variety of things or objects, these objects are equipped with sensing, actuating,

computing, communicating, and storage capabilities allowing them to observe and interact with their environment and cooperate with other objects in order to achieve common goals. These objects could be Radio-Frequency IDentification (RFID) tags, mobile phones, laptops, bracelets, glasses, etc.

These objects will be integrated into complex systems and use sensors and actuators to observe and interact with their physical environment. Such complex systems might include, for example

- food packaging that records temperature throughout the supply chain
- kinematic sensors for assisting rehabilitation
- temperature, humidity and soil moisture sensors for precision agriculture
- air conditioning valves for stabilizing temperature
- solenoid valves for controlling irrigation

### 2.2.1 IoT Applications

IoT is a technology that will allow people and objects in the physical world as well as data and virtual environments to interact with each other so as to create smart environments such as smart transport systems, smart cities, smart health, smart energy, etc., as part of a prosperous digital society. IoT applications will help solve some of the problems that society faces today: remote health surveillance will contribute to the autonomy of the elderly; connected agricultural units will optimize the use of water and fertilizers and so improve agro-industry; connected vehicles will facilitate urban traffic control and reduce pollution and carbon footprints; connected smart grids will help to optimize energy consumption and distribution and the maintenance of electrical infrastructure; etc. These examples illustrate how IoT is likely to improve the quality of people's lives, create new markets and new jobs, increase economic growth and be an impetus for competition.

### 2.2.2 IoT Architecture

The IoT should not be considered as a utopian concept. In fact, it is and will be based on several enabling technologies such as RFID, Near Field Communication (NFC)[3], sensor and actuator networks, Machine-to-machine communications (M2M [31]), the UWB or 3/4G, IPv6, 6LoWPAN and RPL [70], etc. All of which should play an important role in the development of the IoT. The IoT sees its roots back to the M2M (Machine-to-Machine) for remote process monitoring. This technology has evolved to the concept of IoT since the advent of IP mobile cellular networks during the 2000s. ETSI advocates M2M paradigm shift towards IoT. ETSI proposes an architecture based on three domains as shown in Figure 2.1: the domain of network objects, the network access domain, and the field of M2M applications and client applications.

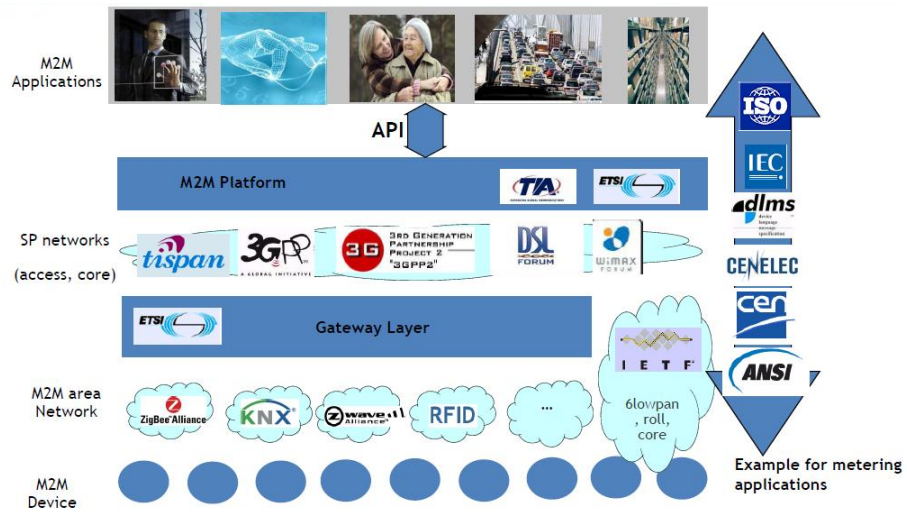



Figure 2.1: IoT/M2M standardization landscape. Source ETSI presentation at MWC 2011[30]

This architecture allows the coexistence of several current and future technologies that enable the development of the Internet of things:

- In the domain of network objects, there are different interconnection technologies such as: M2M [31], RFID, IEEE802.15.4, IETF-6LoWPAN, IETF-RPL [70], etc., and gateways to the transportation backbones.
- In the backbone domain, there are various technologies of network transport and access such as: xDSL, WiMax, WLAN, 3/4G, etc.
- In the field of M2M applications and client applications, we find M2M platforms, M2M applications middleware and APIs, business processes exploiting the Internet of Things, etc.

## 2.3 Internet of Things Security

IoT raises important questions and introduces new challenges for the security of systems and processes and the privacy of individuals. Some IoT applications are tightly linked to sensitive infrastructures and strategic services such as the distribution of water and electricity and the surveillance of bridges and buildings. Other applications handle sensitive information about people, such as their location and movements, or their health and purchasing preferences. Confidence in and acceptance of IoT will depend on the protection it provides to people's privacy and the levels of security it guarantees to systems and processes.

### 2.3.1 IoT Vulnerabilities and Threats

According to the US National Intelligence Council (NIC) IoT is one of the ten most disruptive technologies, with a potentially deep impact on society [53]. The NIC forecasts that by 2025, IoT nodes will reside in everyday things such as food packaging, furniture and documents. Technological advances, combined with strong demand and market opportunities, are predicted to encourage the adoption and deployment of the IoT on a large scale. However, the fear is that everyday objects may become potential security threats. Worse, large-scale penetration by IoT into everyday life will spread these threats much more widely than today's Internet does [53]. Indeed, the ubiquity of IoT is almost certain to increase the number of attacks on data and networks. Moreover, the merging of the physical world and the virtual world that IoT makes possible will open the door to new types of threats that weigh directly on the integrity of the objects themselves and the systems under their control, as well as on the privacy of individuals.

#### Threats against data and networks

The ubiquity of communicating objects in IoT, without physical protection or permanent surveillance, makes them easy targets for hardware and software attacks. Objects might be stolen, counterfeited or corrupted. Without special measures, the data stored inside them would then become accessible, and this includes cryptographic data that might be used to gain access to other sensitive data or allow illicit privilege-escalation in complex hosting systems. Wireless transmissions can also easily fall prey to eavesdropping and denial of service attacks through jamming [71]. Today there exist cryptographic solutions to provide confidentiality, data integrity, authentication and non-repudiation, but these are not necessarily applicable to tiny embedded devices with resource limitations.

#### Threats against privacy

Ubiquitous computing will prevail in IoT, with potentially dozens of communicating objects per person, including inside the private sphere. These objects from the personal domain could be geolocalized, communicate with other objects through spontaneous ad hoc networks, listen to what the person says, film the person and/or the person's environment, and even record the person's heart rate, breathing rhythm, body temperature, and movement! All this raises legitimate questions about the future of this huge mass of personal and sometimes intimate data. Without strict regulation, strict protection of privacy and a high degree of control of objects by users, IoT will not gain acceptance. The ITU's report on the Internet of Things [65] has pointed to these potential threats. It concludes that the protection of privacy should not be limited to technological solutions, but should include legal, market regulation and socio-ethical considerations.

#### Threats against systems and the physical world

IoT will be part of the physical world and complex systems (urban traffic management, control of production lines, control of supply chains and logistics, remote monitoring of patients and disabled people at home, energy distribution, etc.). Malfunctions, denials of service, or byzantine behavior by IoT objects not only undermine the integrity of the virtual world (consisting of data and information), but also directly impact the processes concerned, potentially causing significant collateral damage. For this reason IoT could be a vehicle of choice for hackers seeking adrenaline rushes and for terrorists seeking to create chaos. For instance, in 2009 , a research team from IOActive demonstrated the existence of security vulnerabilities in the devices used in "smart grids" to control energy distribution[52]. The flaw could allow a potential hacker to spread malicious code and cut the supply of electricity to homes. Threats to physical infrastructures, systems and objects are real and require preventive measures to thwart them and remedial measures for containing and preventing their spread whenever threats become actual security attacks.

### 2.3.2 IoT Security and Privacy Challenges

The Internet of Things must be designed for easy use, masking the underlying technological complexity, and for peaceful manipulation preventing security attacks. In IoT any object is potentially connected to the Internet and able to communicate with other objects. This creates new risks related to the confidentiality, authenticity and integrity of data that is sensed, collected and exchanged between objects. The privacy of individuals must be protected to avoid unauthorized identification and localization. As objects become more autonomous and more intelligent, problems relating to privacy are multiplied. Moreover, the strong integration of IoT with the physical world increases control over this world, but also makes it vulnerable to the potentially hazardous actions of the objects.

An urgent prerequisite for securing IoT is the development of efficient security mechanisms for tiny embedded networks with scarce resources. In recent years the need to develop efficient cryptographic systems in terms of resources (energy, memory, processing) has already been felt with the proliferation of tiny embedded networks such as wireless sensor and actuator networks and mobile ad-hoc networks. The advent of IoT, implying the interconnection of a potentially very large number of objects everywhere, accentuates the problem of scarce resources by adding a problem of scalability. There now follows a brief description of a few of the most challenging scientific and technological issues in securing resource-limited networks:

#### Efficient and secure protocols for low-power lossy networks

One of the recommended technologies for the interconnection of the Internet of Things is IPv6, whose main advantage is to be found in the huge capacity offered by 128-bit addressing. IPv6 would satisfy the addressing requirements of a very large-scale IoT with potentially tens of billions

of objects. However, objects' resource limitations, particularly in terms of energy and intermittent connectivity (LLN: Low Power Lossy Networks), make IPv6 difficult to implement. One area being investigated today in relation to LLN communications is an adaptation of IPv6 to these environments through a series of protocols such as 6LoWPAN (IPv6 over Low power Personal Area Networks-RFC4919) and RPL (IPv6 Routing Protocol for Low Power and Lossy Networks) [70]. In addition, resource constraints in the LLN environments that IoT objects inhabit call for new solutions for communication security in order to thwart potential threats. The two IETF groups 6LoWPAN and ROLL are investigating these security issues and working on the development of a security framework specific to LLN environments [64].

#### Efficient cryptography for tiny embedded networks

Current cryptographic algorithms require processing, memory and energy capacities that may simply not be available in tiny, embedded objects. The emergence of a robust, resource-economical cryptography, combined with advanced energy harvesting techniques, will help to address this issue. A number of research works have demonstrated that elliptic curve cryptography [13] provides robust security while requiring fewer resources compared to classical asymmetric cryptography. Recent developments have also shown that energy may in certain circumstances be harvested from an environment of communicating objects, such as in energy harvesting from vibration and movement [27].

#### Efficient and scalable key management for the Internet of Things

The ubiquity of IoT objects, together with the difficulty of providing physical protection and/or permanent surveillance, puts them at risk of physical compromise by intruders. This can have a significant impact if intruders manage to recover the cryptographic keys that are in the memory of corrupted objects and use them to carry out attacks on the network and various other objects. To reduce the impact of this vulnerability resulting from the integration of objects with the physical world, key management must be resilient, in other words tolerant to the presence of compromised objects. Deng et al. [25] define two properties which must be satisfied in the design of a resilient key management scheme: (i) opaqueness, meaning that an adversary should not have the ability to infer other keys used in the network by compromising a small number of objects and (ii) inoculation, meaning that an adversary should not be able to introduce a non-legitimate object into the network through having compromised a small number of existing objects. In addition to such properties that ensure resiliency and given the huge size of IoT and the dynamism of smart objects it is necessary to consider scalability, support of mobility and efficiency of revocation in the design of any key management solution for IoT.

**Efficient authentication and access control**

Authentication and access control are of vital importance to IoT, whose sheer size makes implementing them a challenge. Tim Polk and Sean Turner (IETF security area editors) argue that in addition to scalability issues, the relationships between objects and users, sometimes complex, make access control more difficult [57]. The diversity of user and object identification techniques is another technological obstacle that requires detailed examination.

Given the sensitivity of targeted IoT applications, we will focus in the remaining of this thesis on access control services. Indeed, any hazardous action or access of an object in the IoT sphere may induce a huge damage on the systems under its control. Therefore, insuring efficient and fine grain access control in IoT is of paramount importance to enable a large adoption of this technology.

## 2.4 Access control in IoT

### 2.4.1 Definition

An access control system aims at controlling who (usually called *subject*) can do what (usually identified as *operation* or *right*) on which resource (the *object*) [37], it assigns and checks the permission to a user allowing her (not) to perform some operation on some resource.

More formally, given the sets $S$, $O$ and $R$ that represent respectively:

$S = \{s_i\}$ the set of all subjects in the system,

$O = \{o_j\}$ the set of all object in the system,

$R = \{r_k\}$ the set of all operations or actions envisaged by the system

an access control system is defined by the set of rules $\sum_n (s_i, o_j, r_k)$ where $s_i \in S, o_j \in O, r_k \in R$ for some $i, j, k$.

When designing an access control system for an internet of things environment, some functional parameters must be taken in consideration:

*Delegation support:* a subject can grant access rights to another subject, as well as grant the right to further delegate all or part of the granted rights.

*Access right revocation:* the ability to revoke access rights previously granted without inducing huge overhead.

*Granularity:* Ideally, access control scheme should provide a fine grain access control policy depending on application level requirements.

*Scalability:* The internet of things connects hundred of billions of objects or things, so the access control mechanism must take in count this specificity of IoT.

*Time efficiency:* the access control mechanism should not induce intolerable delays. Especially, the response time must be bounded to the usability expectations.

*Security:* The access control mechanism must be resistant to the different possible attacks in IoT scenario, such as *replay attack*, *denial of service*, *man in the middle*, etc.

### 2.4.2 Classification of Access Control Solutions for IoT

The most common form of access control systems is based on access control lists (ACLs), which consists of assigning access rights to specific subjects. It is known that ACLs are not suitable for the Internet of Things as they become very complex to manage when the number of subjects and resources increases.

Other access control solutions are proposed to overcome the burden of basic ACLs systems. They can be classified into four approaches: Role based access control, Trust based access control, Credential based Access Control, and Context-Aware Access Control (Cf. Figure 2.2).

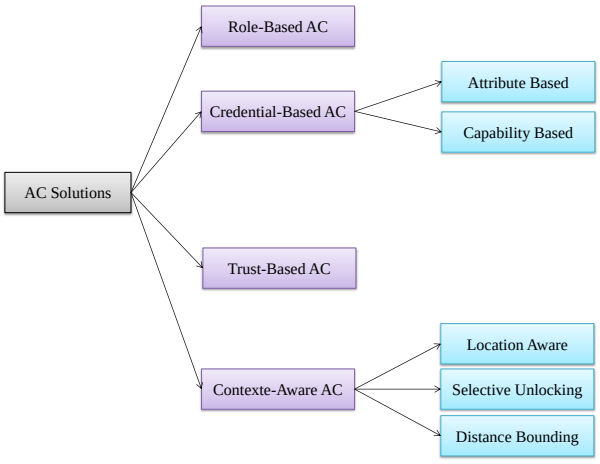



Figure 2.2: Classification of Access Control Solutions

**Role Based Access Control (RBAC):**

In [49], J. Liu et al. propose an authentication and access control approach for the IoT. In the authentication phase, authors used elliptic curve crypto-system with ephemeral private key for establishing a session key for a user and an object. Regarding access control, the authors adopted the RBAC model.

Using RBAC approach leads to roles explosion when the number of resources and/or the number of administrative domains grows.

**Credential based Access Control:** Solutions in this category require a user to have some credentials to gain access to some resource or data, we can cite two subcategories:

- *Attribute-Based Access Control:* a user must have some attributes to be able to access a resource. Giuseppe Bianchi et al. proposed AGREE [12]: an Access control for GREEn wireless sensor networks, which implements the multi-authority version of CP-ABE [11] in energy harvesting wireless sensor network.

- *Capability-Based Access Control:* A capability (known in some systems as a "key") is communicable, unforgeable token of authority, it refers to a value that uniquely references an object along with an associated set of access rights. The capability token grants a process the capability to interact with an object in certain ways.

IACAC [54], CCAAC [5] and CapBAC [37] are some examples of capability based solutions of access control in IoT.

**Trust Based Access Control:** In FTBAC [51], the authors proposed a fuzzy trust based access control approach for the Internet of things. The level of access control from one device to another is directly proportional to the trust it is holding for it. The trust value is related to three components: *experience*, *knowledge* and *recommendation*.

**Context-Aware Access Control:**

The aforementioned solutions are limited when applied to IoT applications, because they are not context aware and thus do not support adaptive access policy definition.

Access control solutions in this category consider the context to decide whether an entity is allowed or not to access a resource or some data. Examples of context elements that can be considered are: location aware [50], motion [62], and distance bounding [17].

- *Location aware access control.* Di Ma et al. proposed two approaches in [50] to improve RFID security and privacy; they take into account contextual information, such as location and speed, sensed by the RFID tag to decide to lock/unlock itself. These approaches are proposed to thwart three well-known attacks: ghost-and-leech attack, reader-and-ghost attack and unauthorized reading. The main contribution of their work is that the two approaches don't require carrying an auxiliary device or necessitate any changes to the current usage model.

- *Context-aware selective unlocking* In this category, we can cite *"Secret Handshake"* [22]. This solution is proposed for accelerometer-equipped RFID tags, using Secret Handshakes, a user must move or shake the tag (or its container) in a particular pattern (parallel movement to the RIFD reader surface for example).

Another solution is named *"Motion Detection"* [62], where a tag would respond only when it is in motion instead of doing so promiscuously. In other words, if the device is still, it remains silent. This approach is not capable of discerning whether the device is in motion due to a particular gesture or because its owner is in motion. Therefore, the false unlocking rate of this approach is high.

- *Distance bounding.* This category of solutions has been used to thwart relay attacks. A distance bounding protocol is a cryptographic challenge-response authentication protocol. Hence, it

requires shared key(s) between tags and readers as other cryptographic protocols. Besides authentication, a distance bounding protocol allows the verifier to measure an upper bound of its distance from the prover [17]. (Normal "non-distance-bounding" protocols are completely ineffective in defending against relay attacks).

### 2.4.3 Discussion

The majority of Access Control systems rely on a server manager that stores all users' access rights in a database (ACL, RBAC, Capability-Based AC, etc.). When a user asks for some data, the server checks database if that user has the permission. If so, the Server Manager grants him the access, otherwise, the user is prevented from accessing the data. In contrast, Attribute-Based Encryption ensures cryptographic Access Control that relies on attributes to define users' secret keys and access policies. Therefore, it allows defining fine-grain access control policies. This is a great advantage in the context of IoT where complex relationships may exist between smart objects and users and therefore require highly expressive access control policies.

## 2.5 Conclusion

In this chapter, we pointed out the sensitivity of Internet of Things to new threats relating to the privacy of users and the reliability of controlled systems. We presented some directions towards IoT security and surveyed particularly access control solutions. Indeed, we showed that access control is a paramount service for IoT given the sensitivity of targeted applications and the damage that may result from hazardous objects actions and access to services and/or data. We showed also that one of the prominent techniques to ensure fine grain access control in IoT is Attribute Based Encryption, given its high expressiveness of access policies. Some challenges, however, are facing the deployment of this technique. In the following chapter we highlight those challenges and survey proposed solutions that pave the way to the adoption of Attribute Based Encryption as a primary technique for ensuring access control in IoT.

# Chapter 3

# Attribute Based Encryption in IoT

## 3.1 Introduction

Attribute-Based Encryption (ABE) is a public key encryption mechanism that allows users to encrypt and decrypt messages based on descriptive user attributes. It is an extremely powerful and promising cryptographic solution to enforce fine-grained access control in the Internet of Things. Using ABE allows keeping encrypted data confidential even if the storage server is untrusted. However, ABE schemes suffer from some drawbacks that break their implementation in the context of IoT. First, the complexity and heavy overhead of ABE schemes make their direct implementation in resource limited environments such as the IoT unsatisfactory. Indeed, the IoT interconnects highly resource-constrained (in terms of energy, storage and computation capability) devices that are not able to run ABE heavy operations (Pairings and exponentiations).

Another challenge facing the implementation of ABE schemes in IoT is the development of a scalable and efficient users' key management mechanism (Attribute/Key revocation). Indeed, existing solutions induce high overhead for each attribute revocation. The side effect induces re-keying and/or re-assignment of attributes to all users.

In this chapter, we present challenges of implementing ABE in the context of IoT. We survey existing solutions allowing to overcome those challenges to pave the way for smooth implementation of ABE schemes in the context of IoT. We also analyze and compare the existing solutions against a set of performance criteria and point out their limitations. In the following chapters we propose our own solutions allowing to further improve implementing ABE schemes in IoT to guarantee fine-grained access control in spite of resource limitations of devices and the huge size of the network.

## 3.2 Attribute Based encryption

### 3.2.1 Overview

Attribute-Based Encryption is a class of Access Control systems that uses attributes to define users' secret keys and access policies. It allows to achieve a cryptographic fine-grained access control.

There are mainly two types of ABE systems: Key-Policy ABE [34] and Ciphertext-Policy ABE [11]. Key-Policy Attribute Based Encryption uses attributes to define the cipher-text access policy, users private keys are associated with an access tree. However, in CP-ABE, users' secret keys are constructed upon a set of attributes defining users' roles in the application, and encryption uses an access tree to define the policy.

At the beginning, the *setup* primitive is executed by the Attribute Authority (AA) to generated a public key $PK$ by formulas 3.1 and 3.7 for CP-ABE and KP-ABE respectively, and a Master Key $MK$ (Formulas 3.2 and 3.8 for CP-ABE and KP-ABE respectively).

The Attribute Authority is a special entity in the system whose role is to manage The universe of attributes and to generate users' private keys. Based on a users' attributes sets (user's access tree for KP-ABE), the Attribute Authority executes the *keygen* primitive for each user and generates a Secret Key $SK$ (Formulas 3.3 and 3.9 for CP-ABE and KP-ABE respectively) for each one of them.

Data are encrypted by running the *encrypt* primitive. Before that, the Data Owner (DO) has to define the access policy to the data he want to encrypt. The access policy is in a form of an access tree (Figure 3.1-(a)). The obtained cipher-text is then publicly shared or stored in cloud based storage servers and it is accessible for everyone. The access control to the plain-text is insured not by the storage server but cryptographically by the access policy defined for it. none other than the authorized users can decrypt the cipher-text.

Figure 3.1-(b) and (c) show three users trying to access to a file encrypted under an KP-ABE and CP-ABE policies respectively. In Figure 3.1-(b), only User 1 and User 2 can decrypt the cipher-text. However, in Figure 3.1-(c) it is User 2 and User 3 who can decrypt the cipher-text.

Contrariwise, The User 2 in Figure 3.1-(b) has an attributes set that does not satisfy the encrypted file access policy, so he cannot decrypt it. The same case with the User 1 in Figure 3.1-(c).

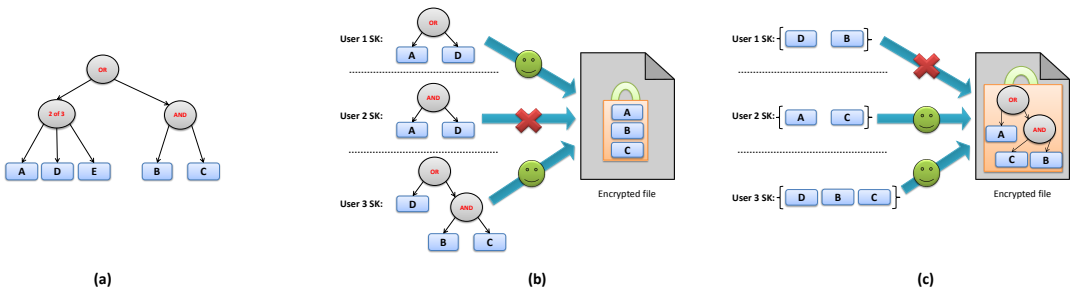



Figure 3.1: (a) Example of access tree. (b) KP-ABE example. (c) CP-ABE example

### 3.2.2 Preliminaries

#### Bilinear maps

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order $p$. Let $g$ be a generator of $\mathbb{G}_0$ and $e$ be a bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. the bilinear map $e$ has the following properties:

1. Bilinearity: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e\left(u^a, v^b\right) = e\left(u, v\right)^{ab}$.
2. Non-degeneracy: $e\left(g, g\right) \neq 1$.

We say that $\mathbb{G}_0$ is a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map $e$ are both efficiently computable. Notice that the map $e$ is symmetric since $e\left(g^a, g^b\right) = e\left(g, g\right)^{ab} = e\left(g^b, g^a\right)$.

#### Access trees

The access tree in KP-ABE scheme defines the access scope of a user's secret key. Each non-leaf node of it represents a threshold gate, described by its children and a threshold value. If $num_x$ is the number of children of a node $x$ and $k_x$ is its threshold value, then $0 < k_x \leq num_x$. Each leaf node $x$ of the tree is described by an attribute and a threshold value $k_x = 1$.

Some functions are defined to facilitate working with access trees:

- *parent(x)*: denotes the parent of the node $x$ in the tree.
- *att(x)*: is defined only if $x$ is a leaf node, and denotes the attribute associated with the leaf node $x$ in the tree.
- *index(x)*: denotes the order of the node $x$ between its brothers. The nodes are numbered from 1 to $num$. (Where $num$ is the number of nodes having the same parent as $x$ including the latter).

**Satisfying an access tree.**
Let T be an access tree with root $r$. Denote by $T_x$ the sub-tree of $T$ rooted at the node $x$. Hence $T$ is the same as $T_r$. If a set of attributes $\gamma$ satisfies the access tree $T_x$, we denote it as $T_x\left(\gamma\right) = 1$. We compute $T_x\left(\gamma\right)$ recursively as follows. if $x$ is a non-leaf node, evaluate $T_{x'}\left(\gamma\right)$ for all children $x'$ of node $x$. $T_x\left(\gamma\right)$ returns 1 if and only if at least $k_x$ children return 1. if $x$ is a leaf node, then $T_x\left(\gamma\right)$ returns 1 if and only if $att\left(x\right) \in \gamma$.

### 3.2.3 ABE schemes

Let $\mathbb{G}_0$ be a bilinear group of prime order $p$, and let $g$ be a generator of $\mathbb{G}_0$. In addition, let $e : \mathbb{G}_0 \times \mathbb{G}_0 \longrightarrow G1$ denote the bilinear map. A hash function $H : \{0,1\}^* \longmapsto \mathbb{G}_0$ will also be used.

### CP-ABE

As mentioned above, CP-ABE [11] is an encryption mechanism where secret keys are constructed upon an attributes sets. Data are encrypted with access policies described with access trees (See Figure 3.1-(c)).

**Primitives.**
There are mainly four primitives defined in the CP-ABE scheme that correspond to system configuration (*setup*), the generation of users' private keys (*KeyGen*), The encryption of sensitive data (*Encrypt*), and finally decryption of encrypted data using private keys (*Decrypt*).

- **setup.** The setup function is run by the Attribute Authority at the bootstrap phase. It takes no input other than the implicit security parameter. It chooses a bilinear group $\mathbb{G}_0$ of prime order $p$ with generator $g$. then it will choose two random exponents $\alpha, \beta \in Z_p$. The public key $PK$ is published as:

$$PK = \left(\mathbb{G}_0, g, h = g^{\beta}, f = g^{1/\beta}, e\left(g, g\right)^{\alpha}\right) \tag{3.1}$$

  and the master key $MK$ is kept secret:

$$MK = (\beta, g^{\alpha}) \tag{3.2}$$

- **keygen(MK, S).** The KeyGen primitive is run by the Attribute Authority for each user joining the system. It takes as input the master key $MK$ and a set of attribute $S$. It chooses a random $r \in \mathbb{Z}_p$, and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$. Then it computes the key as

$$SK = (D = g^{(\alpha+r)/\beta}, \forall j \in S : D_j = g^r H(j)^{r_j}, D'_j = g^{r_j}) \tag{3.3}$$

- **encrypt(PK, M, $\gamma$).** The encryption algorithm takes as input the Public Key $PK$, a message $M$, an access structure $\gamma$ over the universe of attributes. It encrypts the message $M$ under the access tree $\gamma$.

  The algorithm first chooses a polynomial $q_x$ for each node $x$ (including the leaves) in the access tree $\gamma$. These polynomials are chosen in the following way in a top-down manner, starting from the root $R$. For each node $x$ in the tree, set the degree $d_x$ of the polynomial $q_x$ to be one less than the threshold value $k_x$ of that node, that is, $d_x = k_x - 1$.

  Starting with the root node $R$ the algorithm chooses a random $s \in \mathbb{Z}_p$ and sets $q_R\left(0\right) = s$. Then, it chooses $d_R$ other points of the polynomial $q_R$ randomly to define it completely.

For any other node $x$, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses $d_x$ other points randomly to completely define $q_x$.

Let $Y$ be the set of leaf nodes in $\gamma$. The cipher-text is then constructed by giving the access tree $\gamma$

$$CT = \Big(\gamma, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\Big) \quad (3.4)$$

Only users with secret keys satisfying the access tree $\gamma$ can decrypt the cipher-text $CT$.

- **decrypt(CT, SK).** It takes as parameters the cipher-text $CT$ and user secret key $SK$. if the attributes set $S$ upon which $SK$ is constructed satisfies the access policy of $CT$, then the primitive returns the plain-text $M$ of $CT$.

  The algorithm defines a recursive function *DecryptNode(CT,SK, x)* that takes as input a cipher-text $CT = (T_{cpabe}, \tilde{C}, C, \forall y \in Y_{cpabe} : C_y, C'_y)$, a private key $SK$, which is associated with a set $S$ of attributes, and a node $x$ from $T_{cpabe}$. If the node $x$ is a leaf node then we let $i = att(x)$ and define as follows: If $i \in S$, then

$$\begin{aligned} DecryptNode(CT, SK, x) &= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ &= e(g,g)^{rq_x(0)} \end{aligned} \quad (3.5)$$

  if $x$ is a non-leaf node, the algorithm $DecryptNode(CT, SK, x)$ proceeds as follows: For all nodes $z$ that are children of $x$ , it calls $DecryptNode(CT, SK, z)$ and stores the output as $F_z$. Let $S_x$ be an arbitrary $k_x$-sized set of child nodes $z$ such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$. Otherwise, we compute : $F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}$ and obtain after simplification $e(g,g)^{rq_x(0)}$. (More detail can be found in [11]).

  After that, $M$ is obtained this way :

$$M = \tilde{C}/(e(C,D)/A). \quad (3.6)$$

  Where $A = DecryptNode(CT, SK, r) = e(g,g)^{rs}$.

**KP-ABE**

Key-Policy Attribute-Based Encryption [34], in contrast to CP-ABE, uses access trees to specify user's access scope and construct their secret keys, as for the encryption, cipher-texts are encrypted with a set of attributes (See Figure 3.1-(b)).

**Primitives.**

- **setup.** The setup primitive is executed by the attribute authority during the bootstrap phase in order to create the Master Key $MK$ and the Public Key $PK$. It takes no input other than the implicit security parameter.

  It starts by defining the universe of attributes $\mathcal{U} = \{1, 2, ..., n\}$. For each attribute $i \in \mathcal{U}$, a number $t_i$ randomly chosen from $\mathbb{Z}_p$. The primitive chooses also a random number $y$ from $\mathbb{Z}_p$. The Public Key $PK$ and the Master Key are constructed as in formulas 3.7 and 3.8 respectively. The public key $PK$ is published and shared with all entities in the system while the master key $MK$ is kept secret.

$$PK = (T_1 = g^{t_1}, \ldots, T_{|\mathcal{U}|} = g^{t_{|\mathcal{U}|}}, Y = e(g, g)^y). \tag{3.7}$$

$$MK = (t_1, \ldots, t_{|\mathcal{U}|}, y). \tag{3.8}$$

- **keygen($T_{kpabe}$, MK).** The algorithm computes a key that enables the user to decrypt a message encrypted under a set of attributes $\gamma$ if and only if $T_{kpabe}(\gamma) = 1$. It takes a access tree $T_{kpabe}$ and the master key $MK$.

  The algorithm proceeds as follows. First choose a polynomial $q_x$ for each node $x$ (including the leaves) in the tree $T_{kpabe}$. These polynomials are chosen in the following way in a top-down manner, starting from the root node $r$.

  For each node $x$ in the tree, set the degree $d_x$ of the polynomial $q_x$ to be one less than the threshold value $k_x$ of that node ($d_x = k_x - 1$). Now, for the root node $r$, set $q_r(0) = y$ and $d_r$ other points of the polynomial $q_r$ randomly to define it completely. For any other node $x$, set $q_x(0) = q_{parent(x)}(index(x))$ and choose $d_x$ other points randomly to completely define $q_x$. Once the polynomials have been decided, for each leaf node $x$, we give the following secret value to the user: Let $Y_{kpabe}$ be the set of leaf nodes in the access tree $T_{kpabe}$.

$$\forall x \in Y_{kpabe} : D_x = g^{\frac{q_x(0)}{t_i}}.(Where\ \ i = att(x)) \tag{3.9}$$

- **encrypt(M, $\gamma$, PK).** The algorithm takes as inputs a message $M \in \mathbb{G}_2$, a set of attributes $\gamma$, and the public key $PK$.

  It starts by choosing a random value $s \in \mathbb{Z}_p$, then it computes the cipher-text as:

$$E = (\gamma, E' = MY^s, \{E_i = T_i^s\}_{i \in \gamma}) \tag{3.10}$$

- **decrypt(E, D).**

  A recursive algorithm $DecryptNode(E, D, x)$ that takes as input the cipher-text $E = (\gamma, E', \forall i \in \gamma : E_i)$, the private key $D$ (we assume the access tree $T$ is embedded in the private key), and a node $x$ in the tree. It outputs a group element of $\mathbb{G}_2$ or $\perp$. Let $i = att(x)$. If the node $x$ is a leaf node then:

  if $i \in \gamma$,

$$DecryptNode(E, SK_{kpabe}, x) = e(D_x, E_i) = e(g, g)^{sq_x(0)} \tag{3.11}$$

  else, it returns $\perp$.

  We now consider the recursive case when $x$ is a non-leaf node. The algorithm *DecryptNode(E, D, x)* then proceeds as follows: For all nodes $z$ that are children of $x$, it calls *DecryptNode(E, D, z)* and stores the output as $F_z$. Let $S_x$ be an arbitrary $k_x$-sized set of child nodes $z$ such that $F_z \neq \perp$. . If no such set exists then the node was not satisfied and the function returns $\perp$.

  Otherwise, we compute: $F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}$ and obtain after simplification $e(g, g)^{sq_x(0)}$.

  $DecryptNode(E, D, r) = Y^s$ (if and only if the cipher-text satisfies the tree), and we have $E' = MY^s$. The message $M$ is recovered by dividing $E'$ by $Y^s$.

## 3.3 Challenges of implementing ABE in IoT

In this section, we overview some challenges facing the implementation of ABE in IoT. In the following sections we present existing solutions aiming to overcome those challenges. We consider mainly two challenges: (i) resource limitations of smart objects that do not support heavy operations of ABE (pairings and exponentiation), (ii) the size of IoT that makes existing key/attribute revocation techniques non efficient:

### 3.3.1 Resource limitations

Because of its complexity and heavy overhead, ABE scheme left researchers skeptical towards its implementation on small devices like wireless sensors and RFID tags. Indeed, the Internet of

Things interconnects highly resource-constrained devices in terms of energy, storage and computation capability. Such constrained devices are not able to run ABE heavy operations (Pairings and exponentiation). Nevertheless, and given the added value of ABE access control in IoT, many techniques and solutions are proposed to reduce ABE schemes' overhead and/or bypassing the locks through other techniques such as computation offloading, compression, energy harvesting etc.

### 3.3.2 Key/Attribute Revocation

Key Revocation in the context of ABE consists in invalidating the private key of one or many users. Each user's secret key may contain several attributes which can overlap between users. The Attribute Revocation is the mechanism by which one or more attributes are eliminated from the set of attributes of a specific user. This means that the user will no longer be able to decrypt cipher-texts requiring one of the revoked attributes from his attributes set. Both key and attribute revocation are tricky issues [11], as the same attribute could be shared with many other users, and it is very difficult to update a user's key without affecting other users.

## 3.4 Optimized Attribute based Encryption for resource constrained networks

Because of its complexity and heavy overhead, CP-ABE scheme left researchers skeptical towards its implementation on small devices like wireless sensors and RFID tags. That's why many techniques and solutions are proposed to reduce its overhead. We can classify them into four approaches:

- **Constant Cipher-text Size ABE.** Solutions of this category propose constructions of ABE schemes dealing with constant size ciphertexts regardless the number of attributes involved in the encryption. This kind of solutions is pretty interesting as they reduce the storage size of final encrypted messages, especially when we deal with complex and big encryption policies, the size of the final ciphertext will be constant no matter the access policy.

  Emura et al. suggested a scheme with short cipher-texts [28] but access policies are restricted to a single AND-gates. Herranz et al. [39] described a scheme with threshold access policies and constant-size cipher-texts. Yet, their scheme is still not as expressive as one could hope for and it seems difficult to extend it to support general linear-secret-sharing-realizable access structures.

  There are some other solutions like [18], [33], [7], [66], [19], they propose a construction of ABE schemes with a constant cipher-text size. However, many of them are not very expressive they supports only AND-gates in the access tree.

  [8] is an example of constant size ciphertext KP-ABE.

- **Small Secret Key Size.** Constant secret key size [36] [28] (independent of the number of attributes in the key). Regroup many attributes in a single one.

  Attribute Union ABE [19] is another solution proposing to reduce the size of secret key. It gathers many attributes into one attribute leveraging the prime numbers properties. This technique will construct small users' secret keys with a constant size no matter how many attributes a user owns. The problem of this technique is the difficulty to make changes in the scheme to manage the attribute revocation. Moreover, the technique supports only AND-gate in the access policy which makes it less expressive as desired.

- **Computation offloading.**

  The principle of this approach is to rely on one or more powerful trusted devices in the network (ex. remote servers, proxies, assistant nodes) to assist small and constrained devices during cryptographic processes. These assistant devices execute the most consuming operation and offload constrained devices.

  In [45], Yu Jin et al. proposed a Secure and Lightweight CP-ABE (SL-CP-ABE) for mobile cloud computing based on Ibrahimi et al. CP-ABE scheme [42]. They introduced two trusted proxies located in the cloud (Encryption Proxy Server (EPS), Decryption Proxy Server (DPS)) to absorb the overhead of encryption and decryption primitives respectively. However, their scheme still leave some expensive operations to the constrained device, namely: 3 exponentiation operations in encryption and 2 pairings operations in decryption. Moreover, their scheme requires to give a part of user secret key to the cloud which may rise some security and privacy issues.

  Zhou et al. proposed in [75] a Privacy Preserving CP-ABE (PP-CP-ABE) scheme in order to reduce the overhead of CP-ABE [11]. The idea of their solution is to outsource operations of encryption and decryption primitives to an Encryption Service Provider (ESP) and an Decryption Service Provider (DSP) respectively. The disadvantage of their scheme is the lack of flexibility as the access tree must be specified in the form of $T = T_{ESP} \wedge T_{DO}$, where $T_{ESP}$ is the access policy that will be performed by ESP, and $T_{DO}$ is a data owner controlled access policy. In addition, their scheme keep some expensive operations to be executed at the constrained device side (exponentiation and pairing operations).

  In [35], Green Matthew et al. proposed an architecture with a modified ABE scheme that allows reducing the computational load required for decryption on mobile devices by involving a semi-trusted server in the decryption process. They gave new methods for efficiently and securely outsourcing decryption of ABE ciphertexts and showed that it is possible to adapt their outsourcing techniques to both the "Ciphertext-Policy" (CP-ABE) and "Key-Policy" (KP-ABE) types of ABE systems. However, it does not address computational load reduction for the data owner that creates the access policy and the cipher-text. This has inspired Muhammad Asim et al. to propose in [6] a modified CP-ABE allowing to outsource the computational overhead of encryption and decryption phases. Their solution

involves two semi-trusted proxies: cryptographic policy creation is outsourced to a Proxy A, the policy verification is outsourced to another Proxy B. However, their scheme still leaves some expensive operations namely exponentiations.

In [48], Jin Li et al. proposed an ABE system that supports both secure outsourced key-issuing and decryption. Their method offloads all access policy and attribute related operations in the key-issuing process and decryption to a Key Generation Service Provider (KGSP) and a Decryption Service Provider (DSP), respectively. Their construction introduce a trivial policy controlled by a default attribute and use an AND gate connecting the trivial policy and user's policy which seems to be a restriction to the access policy flexibility. However, authors did not propose anything on encryption outsourcing, in addition, the Attribute Authority is usually a powerful server and does not need offloading. Moreover, their seems having some privacy issues against users as the DSP possess a blinded key $\widetilde{TK}$ with the set of user's attribute set.

- **Online/Offline ABE.** This kind of solutions divides some ABE primitives (encryption, decryption) into two parts, the first one with the most overhead is executed online (When the device is connected to an energy source), and the second one which is light is executed offline [41] [23]. These two solutions were developed for mobile devices and they are only applicable if the device is connected periodically to an energy source.

  We can find also in this category AGREE [12]. Bianchi, Giuseppe et al. implemented a multi-authority ABE system [47] in an energy harvesting wireless sensor network. They proposed to exploit the surplus of the energy harvested, that cannot be stored in batteries, to compute some parameters that need lot of computing and which will be used in the near future. These parameters will be stored in the cache memory of the sensor and will be retrieved when it is needed. Nevertheless, their scheme only works for devices that load of energy periodically like devices equipped with an energy harvesting capabilities which must be deployed outdoor.

- **Fast Primitives** In this category, schemes are constructed in order to speed up the running of a specific operation of the scheme by reducing the number of operations to execute. In [40], Susan Hohenberge et al. proposed an Key-Policy Attribute-Based Encryption system with a fast decryption primitive. The decryption process requires only the computation of two (2) pairings regardless the number of attributes involved in the cipher-text. This efficiency is not inconsequential, in fact, there is a significant increase in the private key size by a factor equals to the cardinal of the set of distinct attributes that appear in the private key.

Table 3.1 shows the classification of energy aware ABE systems.

Table 3.1: Comparative table

| | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 |
|---|---|---|---|---|---|---|---|---|---|---|
| KP-ABE/ CP-ABE | [34] | [11] | | | | | | | | |
| Online/ Offline | | | | | | [41] | [23] | [12] | | |
| Computation Offloading | | | | | | [35] | [75] | | [6] [48] | [45] |
| Constant Size Ciphertext | | | | [28] | [39] | [8] [18] | [19] [33] [7] | [66] | | |
| Constant Size Secret Key | | | | [28] | | | [19] | | [36] | |
| Fast Primitives | | | | | | | | [40] | | |

## 3.5 Key/Attribute Revocation in ABE

As we mentioned it in the previous Sections, the Attribute Revocation issue in ABE scheme is very difficult to tackle. Most of the proposed solutions for this problem induce an important overhead that cannot be neglected in the context of the Internet of Things. The difficulty to develop an efficient Attribute Revocation mechanism lies in the fact that one attribute could be shared by multiple users, and every user can have multiple attributes. Thus, revoking a particular attribute for one user will inevitably affect other users sharing the same attribute. The overhead becomes more and more important as the with a high number of users in the system.

This is why developing efficient and scalable attribute revocation mechanism is a very important and challenging task especially for large scale IoT applications.

We can classify the solutions in the literature into three categories (See Figure 3.2):

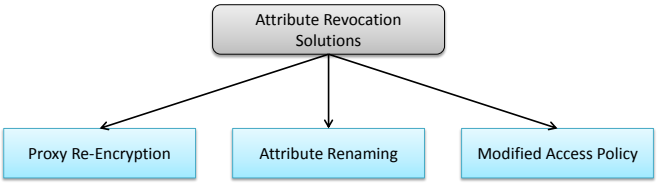



Figure 3.2: Classification of Attribute Revocation solutions for CP-ABE

**Modified Access Policy:**

The idea of this approach is introduced in [11]. As its name indicates it, this solution would require each message to be encrypted with a modified access tree $\gamma'$, which is constructed by augmenting the original access tree $\gamma$ with an additional time attribute $\tau$. The time attribute $\tau$ represents the current 'time period'. Formally, the new access policy $\gamma'$ is as follows: $\gamma' = (\gamma \ \ AND \ \ \tau)$. For example, $\tau$ can be the 'date' (or 'week') attribute whose value changes once every day (every

week). It is assumed that each non-revoked user receives his fresh private key once everyday (or every week) directly from the Attribute Authority.

This solution induces a very important overhead as the access trees are significantly bigger than usual. Indeed, this will enlarge the size of the ciphertext and will increase the computation overhead of the encryption primitive. Moreover, the solution requires every user to possess a time attribute which will increase the size of users' secret keys.

**Attribute Renaming**:

The naive solution is to append to each of the descriptive attributes a date for when the attribute expires. For instance, Pirretti et al. [56] suggest extending each attribute with an expiration date. For example, instead of using the attribute "Computer Science" all short, we might use the attribute "Computer Science: Nov 18, 2017". And when the expiration date comes (Nov 18, 2017), the Attribute Authority renames the attribute "Computer Science" by changing the expired date with the next expiration date: "Computer Science: Jan 20, 2018" for example. Once the Attribute Authority renames that attribute and broadcasts it to all entities in the system, then, it regenerates all secret keys to the non-revoked users (the revocation is materialized by not receiving a new secret key including the renamed attribute).

This type of method has a several shortcomings. Since the Attribute Authority has to generate new secret keys to all non-revoked users whenever the expiration date comes. Thus, users have to maintain a large amount of private key storage and a key for every time period. Further more, this solution induces a heavy overhead as long as all entities will be affected by the revocation.

**Proxy Re-Encryption:**

This approach uses the Proxy Re-Encryption (PRE) technique [14]. A proxy is introduced in the application to absorb the overhead due to the re-encryption. During every decryption process, the proxy is sought to re-encrypt the desired ciphertext in order to be decryptable for authorized users. The proxy is given a proxy key by the Attribute Authority which allows it to re-encrypt ciphertexts without getting any information about the plaintexts.

For example, in [72], Z. Xu et al. addressed user revocation and key refreshing issue for CP-ABE scheme in data-owner-centric environments like those for cloud storage. Their solution, named *DURKR*, uses the Proxy Re-Encryption mechanism and considers only user revocation; it is impossible to revoke a subset of attributes from a user. It requires a cloud storage provider to re-encrypt data for every user request.

Y. Cheng et al. considered in [20] to divide the original data into pieces and encrypt them separately. Once a revoke occurs, the data owner only needs to retrieve and re-encrypt one piece. In this way, the revoked user will have no access to the data. This reduces the workload of re-encrypting the whole data to re-encrypting only pieces of the data.

In [73], Yu et al. tried to resolve the challenging issue of key revocation in CP-ABE by considering practical scenarios like data sharing in which semi-trusted on-line proxy servers are available.

Their solution integrates Proxy Re-Encryption (PRE [14]) technique with CP-ABE and enables the authority to revoke user attributes and to delegate laborious tasks to proxy servers. This solution requires to regenerate all users secret keys and re-encrypting data after every change occurred in the access policy.

In [43], S. Jahid et al. developed a proxy-based revocation solution for attribute based encryption called PIRATTE. The revocation mechanism is based on polynomial secret sharing which allows to perform up to $t$ revocations, where $t$ is the degree of the polynomial in the mechanism. Their solution requires a proxy that participates to the decryption process. Although the scheme achieves dynamic user/attribute revocation without regenerating users keys, it can only revoke up to a predefined numbers of users/attributes.

In [67] and [68], Wang et al. combined Hierarchical identity-based encryption (HIBE) [16] system and CP-ABE to propose a Hierarchical Attribute Based Encryption (HABE) with full delegation. The attribute revocation is achieved by re-encrypting data and updating secret keys. Authors proposed to use Proxy Re-Encryption (PRE) [14] and Lazy Re-Encryption to enhance system performances.

All these solutions have the drawback of requiring an extra entity (proxy or cloud) in the application. This entity is responsible in re-encrypting the message so as authorized users can access to the data. The proxy constitute then a single point of failure. In other words, if the proxy breaks down, the whole system fails. Furthermore, as the proxy is the only one who respond to all users' requests, the scalability of the solution is calling into question as the overhead (computation and storage) at the proxy side becomes important even unbearable.

In table 3.2 we made a comparison between the existing solutions and ours.

| | Re-Encryption | Extra entities (proxies) | Overhead | Renaming Attributes |
|---|---|---|---|---|
| [11] | No | No | Very high | Yes |
| PIRATTE [43] | Yes | Yes | Very high | No |
| DURKR[72] | Yes | Yes | Very high | No |
| [56] | No | No | Very high | No |
| [20] | Yes | No | High | No |
| [73] | Yes | Yes | Very high | No |
| HABE [67],[68] | Yes | Yes | High | No |

Table 3.2: Comparative table

## 3.6 Conclusion

In this chapter, we showed that despite the advantages of Attribute based Encryption in terms of fine-grained access control, its implementation in the context of IoT faces many brakes. After a thorough presentation of attribute based encryption and its two variants CP-ABE and KP-ABE, we discussed challenges relating yo implementing ABE in IoT. In particular we discussed two

main locks: resource constraints in IoT and key/attribute revocation in ABE given the size and dynamics of IoT. Then we surveyed existing solutions aiming to overcome those challenges. We compared existing solutions with respect to performance criteria and pointed out limitations of those existing solutions. In the following chapters we will propose our own solutions pushing further the performances of implementing ABE in IoT in spite of resource constraints of devices and supporting objects dynamism and hence providing efficient key/attribute revocation.

# Chapter 4

# Collaborative ABE schemes in IoT

## 4.1 Introduction

The Internet of Things comprises devices with low capacities in term of energy and computation. This property makes difficult the implementation of cryptographic algorithm like ABE schemes. Indeed, as we pointed it out in the Chapter 2, Attribute Based Encryption (ABE) techniques suffer from the complexity of their schemes and their overhead especially for resource constrained devices. For example, the encryption primitive of CP-ABE requires the computation of $l$ exponentaition (Where )[1].

In this chapter, we present our work on exploiting the heterogeneous nature of the internet of things to propose an implementation of ABE schemes in IoT environments. The basic idea is delegate the most consuming operations to trusted powerful devices in order to offload resource-constrained nodes.

We start this chapter in Section 4.2 with a description of the prerequisites for our collaborative versions of ABE schemes. Network model and assumptions are presented in that introductory section. Section 4.3 and Section 4.4 then detail the proposed approaches Collaborative CP-ABE (C-CP-ABE) and Collaborative KP-ABE (C-KP-ABE) repectively. In Section 4.5, we discuss the security analysis of our collaborative ABE schemes. Section 4.6 discusses the performance analysis of our solutions compared with the original ones. The comparison conducted in terms of number of operations, execution time, communication cost, and energy consumption. In Section 4.7, we discuss the applicability of the computation delegation principle on the CP-ABE/KP-ABE decryption primitives. Finally, Section 4.8 concludes this chapter.

## 4.2 Network model and assumptions

### 4.2.1 Network model

We consider a heterogeneous IoT environment where many devices with different resource capacities coexist in the same network vicinity. We consider three types of devices (See Figure 4.1):

[1]It is well known that the most consuming operations ABE are exponentiation and pairings

- **Highly resource-constrained devices.** These are devices with very limited computation and energy capacities. These devices are usually equipped with reduced processing, storage and energy capacities.

  In this category, we can find RFID tags, smart watches, smart glasses, wireless sensors, etc.

  Devices of this category because of their limited resources are not able to run heavy cryptographic primitives like those of ABE.

- **Unconstrained devices.** Devices of this category are more powerful with less resources limitation. They are generally equipped with powerful processors and are wireline powered. Therefore, they can perform cryptographic primitives easily without draining their energy reserves. Devices of this category can be sensors with energy harvested capability, smart cars, laptops, smart-phones, smart TV, etc.

- **Powerful devices.** This category includes devices having high processing and storage capabilities with unlimited energy source. They are often remote servers that companies make available to users like cloud storage services.

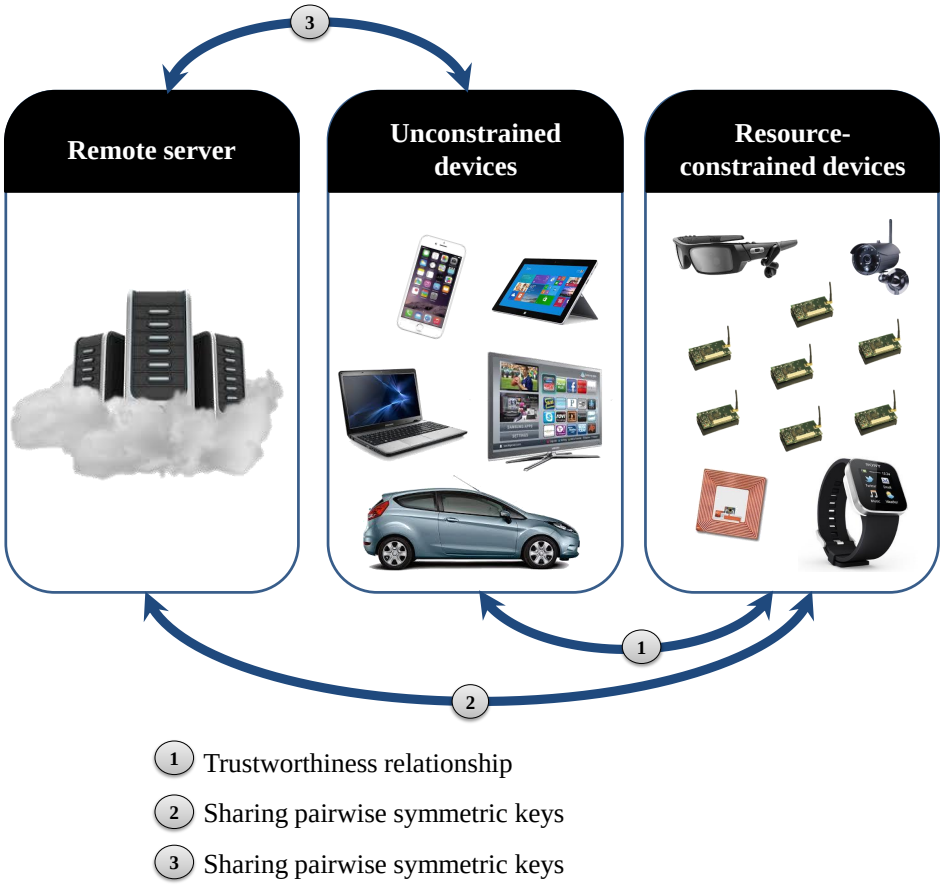



Figure 4.1: Network model

### 4.2.2 Assumptions

In our collaborative solutions, constrained devices will be assisted by unconstrained devices in carrying out ABE operations. Therefore, we assume what follows in our network model:

1. For each resource-constrained device there are at least two trusted unconstrained devices in its neighborhood.
2. Every resource-constrained device shares pairwise keys with two or more unconstrained devices in its neighborhood. These keys may have been generated during a specific bootstrapping phase.
3. Every object in the system shares pairwise keys with the remote servers.

## 4.3 Collaborative CP-ABE

We assume that the universe of attributes is known by all entities of the system and each attribute is identified by and integer. We use the notation $Id(Att)$ to represent the identifier of an attribute $Att$.

Let A be a constrained Data owner. **Node A** aims to encrypt a data under an access $T_{cpabe}$ and send the result to the **Cloud Server**. The server stores encrypted data sent by IoT devices. During the encryption process, node A is helped by a set of trusted **Assistant Nodes**. These assistant nodes are selected from the neighborhood of node A. They execute the exponentiation operations instead of Node A.

Figure 4.2 overviews of our collaborative CP-ABE encryption primitive.

### 4.3.1 Per-phase Collaborative Encryption Primitive

1. Phase 1: The resource-constrained Data Owner which has to encrypt a message $M$ (Device A in Figure 4.2), starts by choosing $p$ trusted unconstrained nodes from his neighbors. These nodes (Assistant Nodes) will assist the Data Owner during the encryption process. Notice that the existence of these trusted nodes is assumed in the network model (Section 4.2.2).
2. Phase 2: In this phase, the Data Owner defines the access policy for the message $M$ and constructs the corresponding access tree $T_{cpabe}$. After that, it generates the polynomials $q_y$ (for all $y \in Y_{cpabe}$ ) associated with nodes of $T_{cpabe}$ as described in the encryption algorithm in the subsection 3.2.3.

   Now, the Data Owner splits $s$ ($s$ represents $q_R(0)$) into $p$ parts $s_i$ (each part is destined to an Assistant Node), such that the sum of all $s_i$ gives $s$ (Equation 4.1). Likewise, it splits

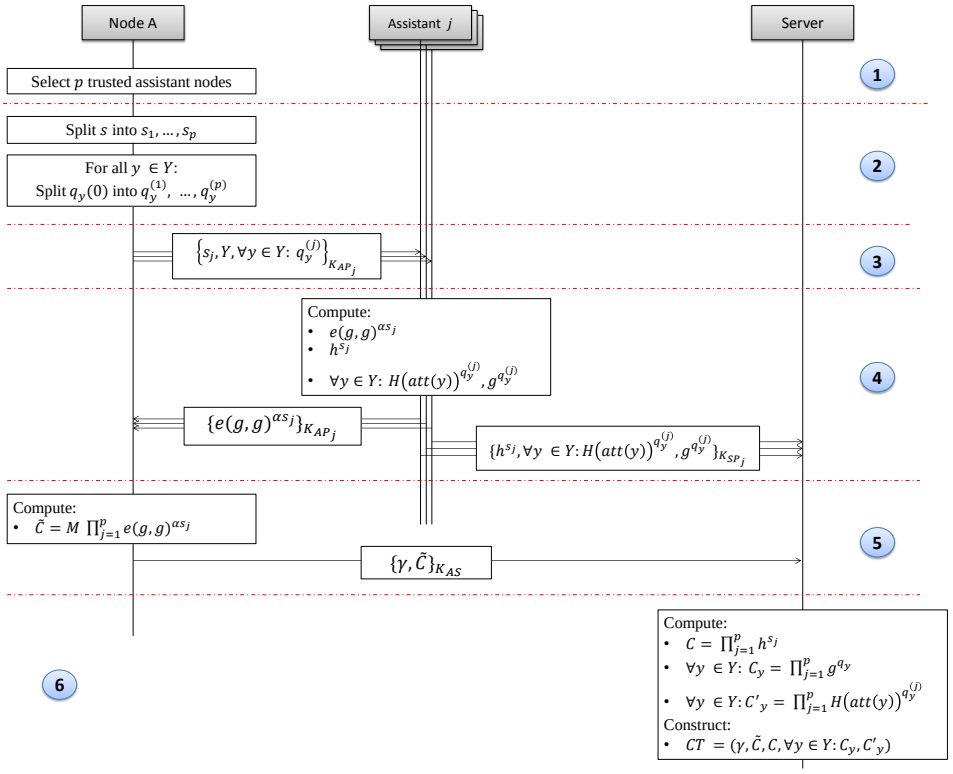



Figure 4.2: C-CP-ABE Scheme

every $q_y(0)$ into $p$ parts $q(i)_y$ for all $y \in Y_{cpabe}$ (Equation 4.2), where $Y_{cpabe}$ is the set of leaf nodes of the access tree $T_{cpabe}$.

$$s = \sum_{j=1}^{p} s_j. \tag{4.1}$$

$$\forall y \in Y_{cpabe} : q_y(0) = \sum_{j=1}^{p} q_y^{(j)}. \tag{4.2}$$

3. Phase 3: To each assistant node $AN_i$ (Where $i = 1, \cdots, p$), the Data Owner securely sends $s_i$, and for all $y \in Y$ , it sends $Id(att(y))$ and $q(i)_y$. These information are transmitted encrypted with the shared key between A and the assistant node $AN_i$ (Assumption 2).

4. Phase 4: Each assistant node $AN_i$ computes $e(g,g)^{\alpha s_i}$ (Formula 4.3), $h_i^s$ and sends it back to the Data Owner after encryption, it computes also $H(att(y))q(i)y(0)$ and $gq(i)y(0)$ for all $y \in Y$ and sends them encrypted to the server using the shared key between them (Assumption 3). We remind that $h, e(g,g)^{\alpha}$ and $g$ are parts of the public key $PK$, therefore the Data Owner has not so send them in the previous phase.

$$e(g,g)^{\alpha s_i} : q_y(0) = \sum_{j=1}^{p} q_y^{(j)}. \tag{4.3}$$

5. *Phase 5:* Device A uses the intermediate results calculated by the assistant nodes to compute the element $\tilde{C}$ of $CT$ this way:

$$\begin{aligned} \tilde{C} &= M \prod_{i=1}^{p} e(g,g)^{\alpha s_i} \\ &= M e(g,g)^{\alpha \sum_{i=1}^{p} s_i} \\ &= M e(g,g)^{\alpha s} \end{aligned} \tag{4.4}$$

After that, Device A encrypts both $\tilde{C}$ and $T_{cpabe}$ using the symmetric key $K_{AS}$ shared with the remote server and sends him the result.

6. *Phase 6:* The remote server receives intermediate results from assistant nodes and decrypts them. It also receives from device A the access policy and $\tilde{C}$ after decryption. The server uses the intermediate results to compute the elements: $C$, and for all $y \in Y_{cpabe}$ : $C_y$ and $C'_y$ using formulas 4.5, 4.6 and 4.7.

$$C = \prod_{i=1}^{p} h^{s_i} = h^{\sum_{i=1}^{p} s_i} = h^s \tag{4.5}$$

$$\begin{aligned} \forall y \in Y_{cpabe} : C_y &= \prod_{i=1}^{p} g^{q_y^{(i)}} \\ &= g^{\sum_{i=1}^{p} q_y^{(i)}} \\ &= g^{q_y(0)} \end{aligned} \tag{4.6}$$

$$\begin{aligned} \forall y \in Y_{cpabe} : C'_y &= \prod_{i=1}^{p} H(att(y))^{q_y^{(i)}} \\ &= H(att(y))^{\sum_{i=0}^{p} q_y^{(i)}} \\ &= H(att(y))^{q_y(0)} \end{aligned} \tag{4.7}$$

After that, it simply constructs the cipher-text $CT$ by combining all computed results using formula 4.8.

$$CT = \left(T_{cpabe}, \tilde{C}, C, \forall y \in Y_{cpabe} : C_y, C'_y\right) \tag{4.8}$$

## 4.4 Collaborative KP-ABE

### 4.4.1 Per-phase Collaborative Encryption Primitive

Let A be a constrained node. **Node A** aims to encrypt a data under a list of attributes $\gamma$ and send the result to the **Cloud Server**. The server stores encrypted data sent by IoT devices. During the encryption process, node A is helped by a set of trusted **Assistant Nodes**. These assistant nodes are selected from the neighborhood of node A. They execute the exponentiation operations instead of Node A.

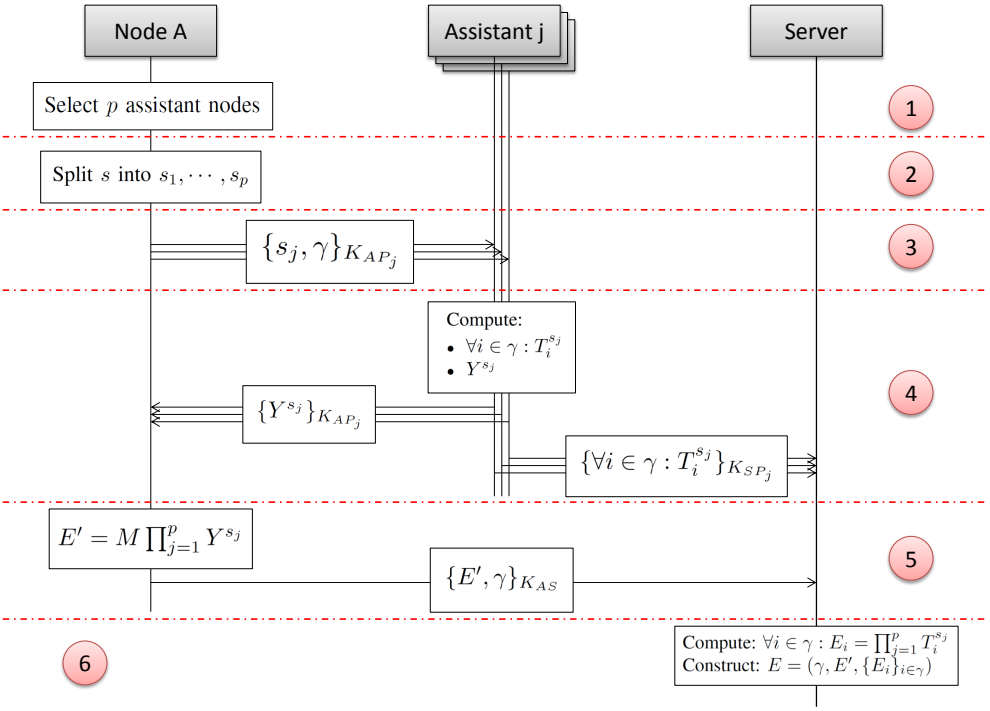



Figure 4.3: C-KP-ABE Scheme

Figure 4.3 shows node A in the left, cloud server on the right, and assistant nodes in the middle. It illustrates the different steps of our Collaborative KP-ABE encryption process. These steps are detailed bellow.

1. Phase 1: The Data Owner selects $p$ trusted nodes among those situated in its neighborhood, these nodes (assistant nodes) will assist the Data Owner during the encryption process. Notice that the existence of such trusted nodes is assumed in the network model (Section 4.2.2).

2. Phase 2: First, the Data Owner defines the set of attributes $\gamma$ that will be associated to the ciphertext $E$. Then, it chooses a random value $s \in \mathbb{Z}_p$ and splits it into $p$ values $s_j$ such that the sum of all these values $s_j$ gives $s$ (Formula 4.9).

$$s = \sum_{j=1}^{p} s_j. \tag{4.9}$$

3. Phase 3: The Data Owner sends each $s_j$ to an assistant node $AN_j$ along with the attributes set $\gamma$, all encrypted with the shared symmetric key $K_{AP_j}$ with the latter. The value of $Y$ doesn't require to be sent because it is a part of the public key $PK$, so all the assistant nodes already know it.

4. Phase 4: After receiving $s_j$, the assistant node $AN_j$ computes $Y^{s_j}$, and for all $i \in \gamma : T_i^{s_j}$. Then, it sends back $Y^{s_j}$ to the Data Owner (device A), and hands over all $\{T_i^{s_j}\}_{i\in\gamma}$ to the remote server after encryption with the shared key.

5. Phase 5: When the Data Owner receives the interim results from assistant nodes, it computes $E'$ as follows:

$$\begin{aligned} E' &= M \prod_{j=1}^{p} Y^{s_j} \\ &= MY^{\sum_{j=1}^{p} s_j} \\ &= MY^{s}. \end{aligned} \tag{4.10}$$

   After that, the Data Owner sends this result along with the attributes set $\gamma$ to the remote server all encrypted by the symmetric shared key $K_{AS}$ with the latter.

6. Phase 6: The remote server receives interim results from assistant nodes and computes $\{T_i^s\}_{i\in\gamma}$ elements.

$$\begin{aligned} \forall i \in \gamma : E_i &= \prod_{j=1}^{p} T_i^{s_j} \\ &= T_i^{\sum_{j=1}^{p} s_j} \\ &= T_i^{s} \end{aligned} \tag{4.11}$$

   It receives also from the Data Owner the attributes set $\gamma$ and $E'$. It constructs the ciphertext as follows:

$$E = (\gamma, E', \forall i \in \gamma : E_i = T_i^s) \tag{4.12}$$

The cipher-text $E$ will henceforth be available at the cloud server to whoever requests it.

## 4.5 Security Analysis

### 4.5.1 Collaborative CP-ABE

To decrypt a ciphtertext $CT$, a user must possess a valid secret key $SK$ associated with an attributes set $S$ satisfying the access tree $T_{cpabe}$ in $CT$ ($T_{cpabe}(S) = 1$). To decrypt a message, a user clearly must recover $e(g,g)^{\alpha s}$ by executing the function $DecryptNode(CT, SK, r)$. Authors gave in [11] an analysis of the impossibility of computing $e(g,g)^{\alpha s}$ by an unauthorized attacker. The random values used in generating secret keys and encrypting messages prevent any user without the necessary attributes to compute $e(g,g)^{\alpha s}$.

Another way to decrypt the message, without having the necessary attributes, is to calculate/guess the value of $s$ used for encryption by the Data Owner. An attacker could try eavesdropping the communication between the Data Owner and the Assistant Nodes during the Phase 2 (see Section 4.3). But it cannot recover the value of $s$ because of secure communication between the Data Owner and the assisting nodes thanks to the Assumption 2 (see Section4.2.2), and thanks to the trustworthiness of the Assistant Nodes (Assumption 1 and 2), they will not collide to recover the value of $s$.

It is important to notice that even the storage server could not recover the original message. Indeed, the intermediate results received from the assistant nodes are not sufficient to recover the value of $s$.

### 4.5.2 Collaborative KP-ABE

To decrypt a cipher-text $E$, a user must have a valid secret key $SK$ whose corresponding access tree $T_{kpabe}$ satisfies the attributes set $\gamma$ associated with $E$. The authorized user recovers the value of $Y^s$ using her/his secret key $SK$ and $E_i$. After that, she/he can get $M$ by a simple division of $E'$ by $Y^s$ [34]. Therefore, an unauthorized user can recover the plaintext $M$ if only she/he knows $Y^s$ or even the random number $s$ used to encrypt the data. Otherwise she/he must have a valid secret key to do so.

An unauthorized user who eavesdrops the communications cannot get any information about $Y^s$ or $s$ during the collaboration of the Data Owner, assistant nodes, and the storage server. Because all the communications are encrypted using symmetric shared keys (Assumptions 2 and 3 in SectionChapter4-Requirements-Assumptions).

Assumption 1 presented in section 4.2.2 ensures that assistant nodes will not disclose any information exchanged with the Data Owner or even collide between them to compute $s$ or $Y^s$ using $s_i$ or $Y^{s_i}$ ($i \in 1, \cdots, p$) respectively. Indeed, assistant nodes are assumed to be trusted. Assumption 1

also ensures that assistant nodes will not give erroneous results or abstain from participating to the encryption process.

The remote server also cannot recover the plaintext of the encrypted data. The information received from the assistant nodes and the Data Owner do not allow computing $s$ or $Y^s$.

To prevent our solution from *replay attacks* we can use a nonce at the beginning of the encryption process. So as, attackers cannot reuse old communications.

## 4.6 Performance Analysis

In this section, we present the analysis of performances of our collaborative versions of both CP-ABE and KP-ABE. We start by giving the experiments setting and simulation model. Then, we compare our solutions to the original ones in terms of overhead, execution time, communication cost, and energy consumption.

### 4.6.1 Experiments Settings

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$ be three multiplicative cyclic groups of prime order $p$. Let $g$ be a generator of $\mathbb{G}_1$ and $e$ be a bilinear map, $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$.

Table 4.1 presents all group elements sizes for different pairings parameters [4]. We analyze our solutions for these four pairings parameters and we show their performances in each case.

Table 4.1: Size of the elements in different pairings

| | a | a1 | d159 | f |
|---|---|---|---|---|
| $\mathbb{G}_1$ (bytes) | 132 | 264 | 44 | 44 |
| $\mathbb{G}_2$ (bytes) | 132 | 264 | 124 | 84 |
| $\mathbb{G}_t$ (bytes) | 132 | 264 | 124 | 244 |
| $\mathbb{Z}_p$ (bytes) | 24 | 132 | 24 | 24 |

### 4.6.2 Number of Operations

In this section, we discuss the performance of our collaborative versions of CP-ABE and KP-ABE in terms of number of operations (Multiplication and Exponentiation operations) to be computed during the encryption process by the Data Owner. It is well known that the cost of one multiplication is negligible compared to the cost of an exponentiation in terms of both execution time and energy consumption [11].

**Collaborative CP-ABE Encryption vs. CP-ABE Encryption**

Let $Y_{cpabe}$ be the set of leaf nodes in the access tree $\gamma$, and $p$ be the number of chosen Assistant Nodes to assist the Data Owner during the encryption process.

Table 4.2 shows the comparison between our collaborative CP-ABE and the original one in terms of number of operations to be executed during the encryption process. As we stated above, the data owner is resource-constrained device, and our solutions aims to reduce its overhead. We notice that our collaborative CP-ABE eliminates the overhead due to exponentiation operations while it should compute $2 + 2|Y_{cpabe}|$ operations in original CP-ABE.

Table 4.2: Computation comparison

| | | Number of Multiplications | Number of Exponentiations |
|---|---|---|---|
| CP-ABE | Data Owner | 1 | $2 + 2\|Y_{cpabe}\|$ |
| | Remote server | 0 | 0 |
| | Assistant node | - | - |
| C-CP-ABE | Data Owner | $p$ | 0 |
| | Remote server | $2 + 2\|Y_{cpabe}\|$ | 0 |
| | Assistant node | $(p-1) + (2p-2)\|Y_{cpabe}\|$ | 0 |

**Collaborative KP-ABE Encryption vs. KP-ABE Encryption**

Let $\gamma$ be the attributes set with which the data is encrypted, and $p$ be the number of selected assistant nodes to assist the Data Owner. Table 4.3 shows the number of operations (multiplication and exponentiation) executed during the encryption process by the different parties (Remote server, Assistant node, Device A) in both cases (Original KP-ABE and collaborative KP-ABE).

In C-KP-ABE, the Data Owner has not to compute any exponentiation, whereas it has to compute $|\gamma| + 1$ exponentiations in the original KP-ABE.

Table 4.3: Computation comparison

| | | Number of Multiplications | Number of Exponentiations |
|---|---|---|---|
| KP-ABE | Data Owner | 1 | $\|\gamma\| + 1$ |
| | Remote server | 0 | 0 |
| | Assistant node | - | - |
| C-KP-ABE | Data Owner | $p$ | 0 |
| | Remote server | $\|\gamma\|(p-1)$ | 0 |
| | Assistant node | 0 | $\|\gamma\| + 1$ |

From Table 4.2 and Table 4.3, we notice that the overhead due to the exponentiations is displaced from resource-constrained Data Owner to more powerful devices: Assistant Nodes and the Remote Server with a slight increase of the number of multiplications to be computed.

### 4.6.3 Execution time cost

In this section, we are interested in the execution time of the encryption primitive at the data Owner side. We point out that we do not include the communication time due to the exchanges between Data Owner and Assistant Nodes, and Data owner and remote server.

The experiments are executed on a Raspberry PI 2 running a Ubuntu 15.10 installed in a 8GB SD card. PBC [4] version 0.15.14 and GMP [2] version 6.1.0 are used to implement our solutions and compare them to original ones.

**Collaborative CP-ABE Encryption vs. CP-ABE Encryption**

Table 4.4 shows the comparison between CP-ABE and our Collaborative CP-ABE with five assistant nodes ($p = 5$) in term of execution time of the encryption primitive. During the experimentation, we used different access trees with different sizes ($1 \leq |Y_{cpabe}| \leq 50$) and different pairing parameters. The results of experimentation are also illustrated in Figure 4.4.

We notice that our solution is largely more efficient then the original CP-ABE especially when the number of leaf nodes in the access tree increases.

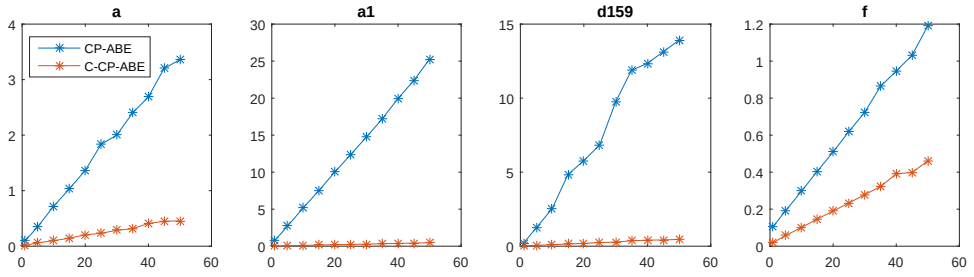



Figure 4.4: Comparison between CP-ABE and C-CP-ABE in term of execution time ($p = 5$)

**Impact of the number of assistant nodes for Collaborative CP-ABE.**

We have also conducted an experimentation in order to see the impact of the number of assistant nodes $p$ on the time execution for different pairing parameters.

Table 4.5 and Figure 4.5 shows the results of the experimentation. We remark that the time execution increases with higher values of $p$. This is due to the Phase 2 (See Section 4.3.1) when the Data Owner has to split the number $r$ and all $q_y(0)$ ($y \in Y_{cpabe}$).

The time spent during the Phase 5 (See Section 4.3.1) is lower and almost negligeable.

**Collaborative KP-ABE Encryption vs. KP-ABE Encryption**

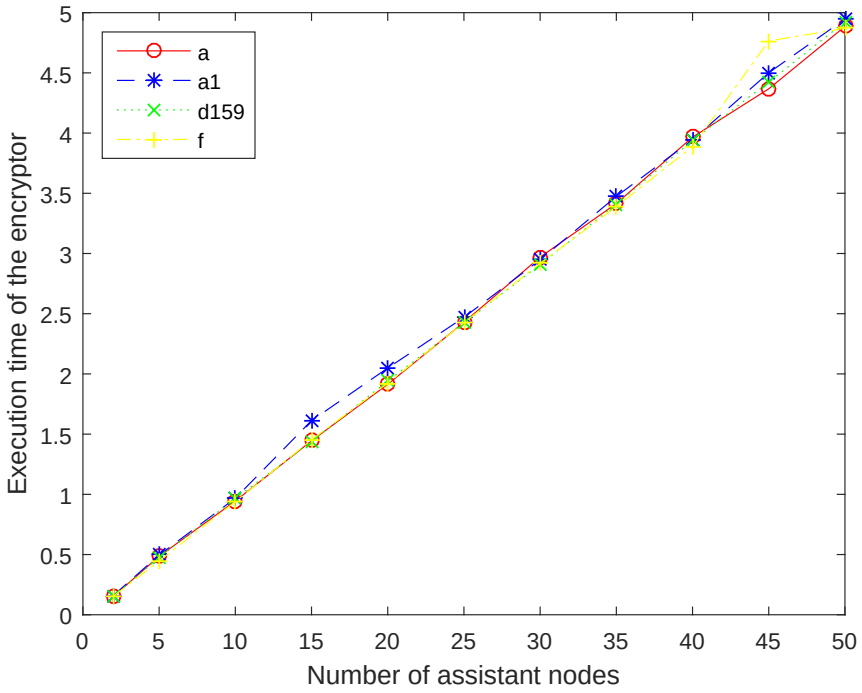



Figure 4.5: Impact of the number of assistant nodes in C-CP-ABE

Table 4.6 shows the comparison between KP-ABE and our Collaborative KP-ABE with five assistant nodes ($p = 5$) in term of execution time of the encryption primitive. During the experimentation, we used different attributes sets with different sizes ($1 \leq |\gamma| \leq 100$) and different pairing parameters ("a", "a1", "d159", and "f"). The results of experimentation are also illustrated in Figure 4.6.

We notice that our solution is largely more efficient then the original KP-ABE especially when the number of attribute in $\gamma$ increases.

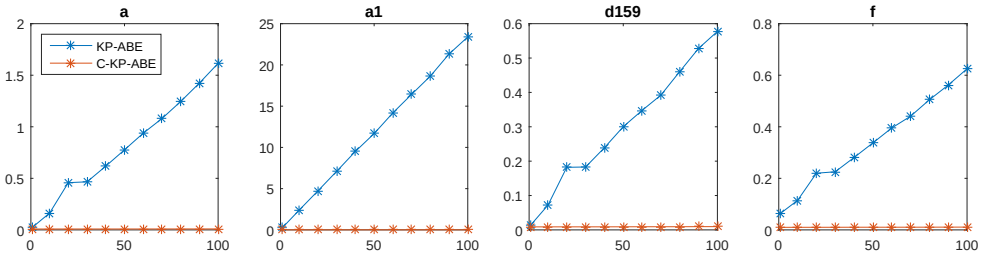



Figure 4.6: Comparison between KP-ABE and C-KP-ABE in term of execution time ($p = 5$)

**Impact of the number of assistant nodes for Collaborative KP-ABE.**

From Table 4.7 and Figure 4.7, we notice that the execution time at the side of the Data Owner increases for higher number of assistant nodes. This is due to the number of multiplications to be executed after receiving partial results from assistant nodes (Section 4.4.1- Phase 5).

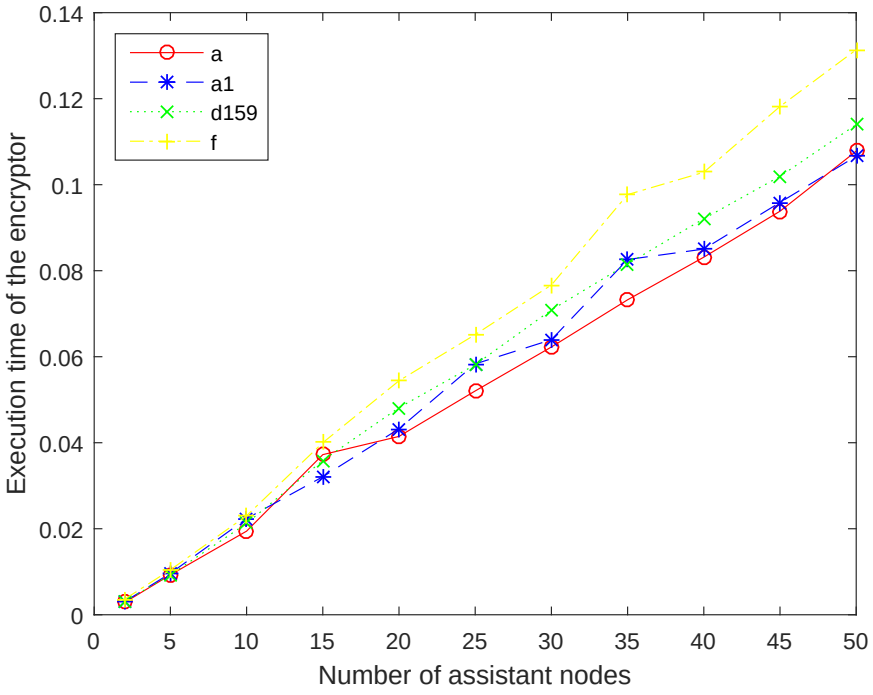



Figure 4.7: Impact of the number of assistant nodes in C-KP-ABE

### 4.6.4 Communication cost

In original version of CP-ABE and KP-ABE, device A has only to send the final cipher-text to the remote sever (cloud server). However, in our collaborative versions (C-CP-ABE and C-KP-ABE), Data Owner exchanges many messages with assistant nodes and the remote server.

We recall that (for CP-ABE):

- $D, D'_j, C_y, C \in \mathbb{G}_1$
- $D_j, C'_y \in \mathbb{G}_2$
- $\tilde{C} \in \mathbb{G}_T$
- $s, s_i, q_y(0) \in \mathbb{Z}_p$

and (for KP-ABE):

- $E_i \in \mathbb{G}_2$
- $Y \in \mathbb{G}_T$

In this section, we compare the cost due to communications in terms of consumed energy by the Data Owner during the encryption process between CP-ABE and Collaborative CP-ABE, and between KP-ABE and Collaborative KP-ABE. We adopt the model of LEACH proposed in [38].

***LEACH Energy Model [38]:***

The amount of energy needed by the radio to transmit a $l$-bit message a distance $d$ is given with formula 4.13.

$$\begin{aligned} E_{Tx}(l,d) &= E_{Tx-elec}(l) + E_{Tx-amp}(l,d) \\ &= lE_{elec} + l\epsilon_{fs}d^2 \end{aligned} \tag{4.13}$$

Where $E_{Tx-elec}(l)$ represents the electronics energy and $E_{Tx-amp}(l,d)$ is the amplifier energy. $d$ represents the distance between the transmitter and the receiver which is smaller then the threshold $d_0$ as we have assumed that all assistant nodes are situated in the neighborhood of Node A. $d$ is set to $100m$. The communication energy parameters are set as: $E_{elec} = 50nJ/bit$, $\epsilon_{fs} = 10pJ/bit/m^2$. To receive a message, the radio expends:

$$\begin{aligned} E_{Rx}(l) &= E_{Rx-elec}(l) \\ &= lE_{elec} \end{aligned} \tag{4.14}$$

**Collaborative CP-ABE Encryption vs. CP-ABE Encryption**

Based on LEACH energy model, we determine with respect to the number of leaf nodes $|Y|$ the maximum number of assistant nodes that can be sought during the encryption process such that C-CP-ABE communication cost stays lower than original CP-ABE one. The obtained results for different pairings parameters are summarized in Figure 4.8.

We notice that choosing the configuration in the file "a" of PBC library [4] gives the best results. For example, for an access tree with 10 leaf nodes ($|Y| = 10$), we could use our C-CP-ABE and solicit 7, 4, 3, and 3 assistant nodes in case of pairing parameters: "a", "d159", "f", and "a1" respectively, and be gainer in communication performances, without considering the computation cost.

**Collaborative KP-ABE Encryption vs. KP-ABE Encryption**

Figure 4.9 illustrates for each pairings parameter the maximum number of assistant that can be sought during encryption without losing in communication performances.

We notice that the gain in communication cost is more interesting with C-KP-ABE than in C-CP-ABE. Indeed, for an encryption using ten (10) attributes, we could solicit assistant nodes in case of pairing parameters: "a", "d159", "f", and "a1" respectively, , and be gainer in communication performances too.

### 4.6.5 Energy consumption

In this section, we try to consider the energy consumption as a metric in the comparison of our solutions and original ones. It includes the energy consumed due to both of computation and

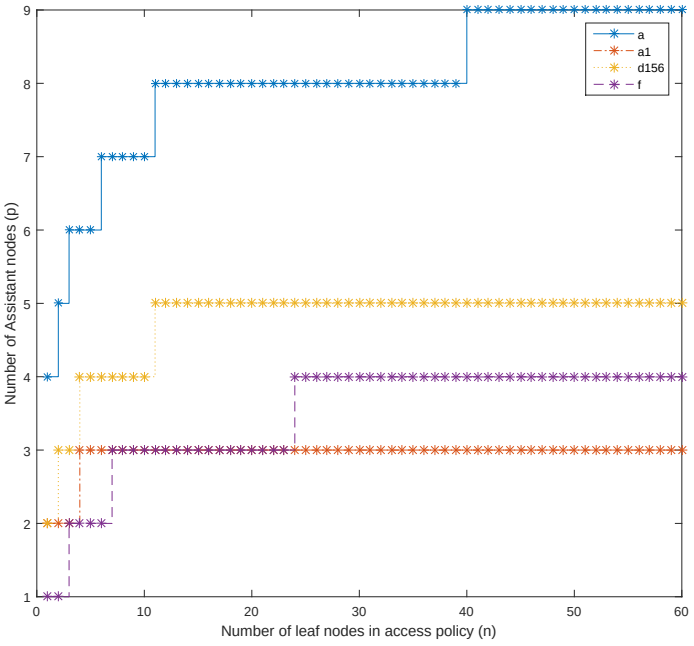



Figure 4.8: Maximum number authorized of assistant nodes for C-CP-ABE without losing in communication performances

communication. This metric is more precise compared to the execution time, as the latter considers only the computation cost.

The result are obtained using PowerTOP [1] which is a Linux tool to diagnose issues with power consumption and power management.

The resource-constrained device is represented by a computer with an IntelÂ® Core$^{TM}$ i5-560M Processor (2.66 GHz with Turbo Boost up to 3.20 GHz*1). The WiFi connection used is of type IEEE 802.11b/g/n*6.

**Collaborative CP-ABE Encryption vs. CP-ABE Encryption**

Table 4.8 and Figure 4.10 show the results of comparison between CP-ABE and C-CP-ABE in terms of energy consumption with the variation of the number of assistant nodes $p$. The number of attributes is set to 5 ($|Y_{cpabe}| = 2$)

We notice that our solution shows better results than original CP-ABE for values of $p$ lower than 15. For values of $p$ greater than 15, our solution consumes more energy than CP-ABE.

Table 4.9 and Figure 4.11 show the results of comparison between CP-ABE and C-CP-ABE in terms of energy consumption by varying the number of attributes in $Y_{cpabe}$ from 5 to 100. The number of assistant nodes is set to 2 ($p = 2$).

We notice that both techniques consumption grows with high values of $|Y_{cpabe}|$. Nevertheless, our solution distinctly outperforms original CP-ABE.

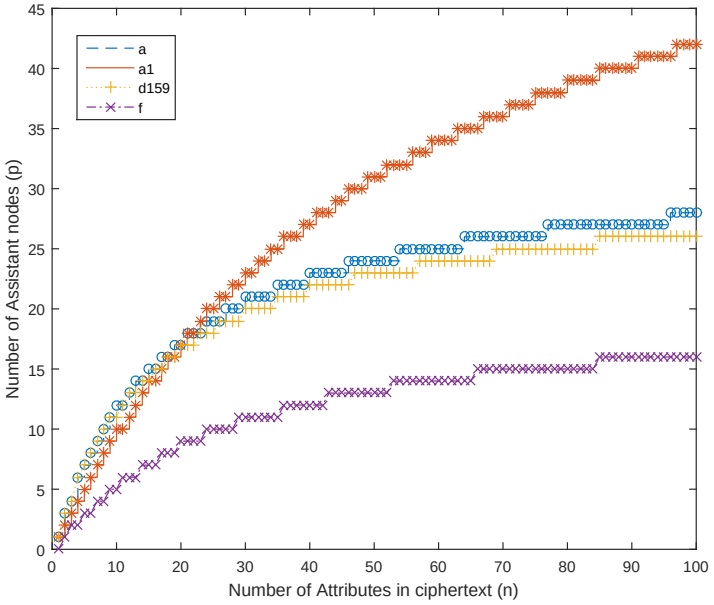



Figure 4.9: Maximum number authorized of assistant nodes for C-KP-ABE without losing in communication performances

#### Collaborative KP-ABE Encryption vs. KP-ABE Encryption

Table 4.10 and Figure 4.12 show the results of comparison between KP-ABE and C-KP-ABE in terms of energy consumption with the variation of the number of assistant nodes $p$. ($|\gamma| = 5$).

We notice that our solution shows better results than original KP-ABE for values of $p$ lower than 20. For values of $p$ greater than 20, our solution consumes more energy than KP-ABE.

Table 4.11 and Figure 4.13 show the results of comparison between KP-ABE and C-KP-ABE in terms of energy consumption by varying the number of attributes in $\gamma$ from 5 to 100. The number of assistant nodes is set to 2 ($p = 2$).

We notice that both techniques consumption grows with high values of $|\gamma|$. Nevertheless, our solution distinctly outperforms original KP-ABE.

## 4.7 Applying delegation to the decryption primitives

In this section, we discuss the applicability of the computation delegation principle on the CP-ABE/KP-ABE decryption primitives. We show how it is possible to delegate exponentiation and pairing operations[2] to assistant nodes. The decrypt-or node will execute multiplications rather than pairing and exponentiation operations.

[2]These are the most consuming operations in CP-ABE/KP-ABE decryption primitives.

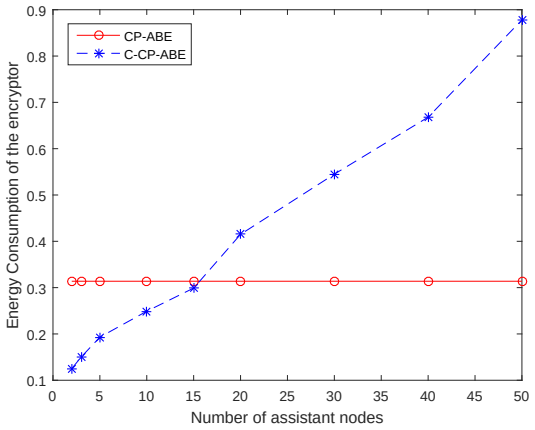



Figure 4.10: Energy consumption of CP-ABE and C-CP-ABE with respect to the number of assistant nodes $p$

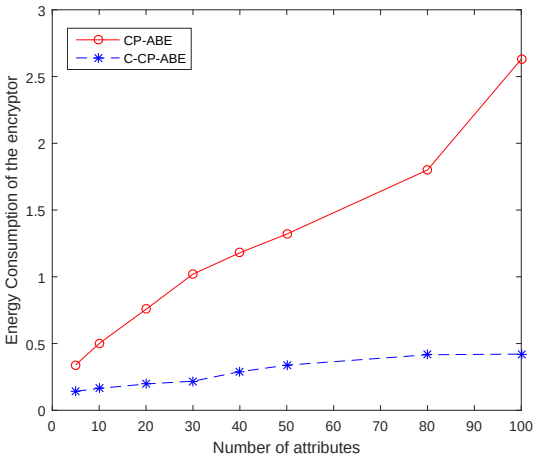



Figure 4.11: Comparison in terms of Energy consumption with respect to number of attributes in $Y_{cpabe}$

### 4.7.1 Principle

In this section, we show the principle of applying computation offloading of decryption primitive. By analyzing the decryption primitive, we notice that, the most consuming operations are exponentiations and pairings. In previous sections, we detailed how it is possible to distribute and delegate exponentiations (See Section 4.3 and Section 4.4). Now, we exhibit how to distribute the computation of a pairing operation.

Let $e$ be a bilinear map, $C$ and $D$ are elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Let $g$ be a generator of $\mathbb{G}_2$.

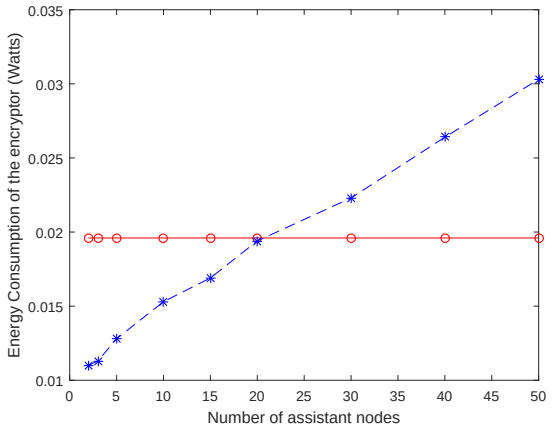



Figure 4.12: Comparison in terms of Energy consumption with respect to the number of assistant nodes $p$

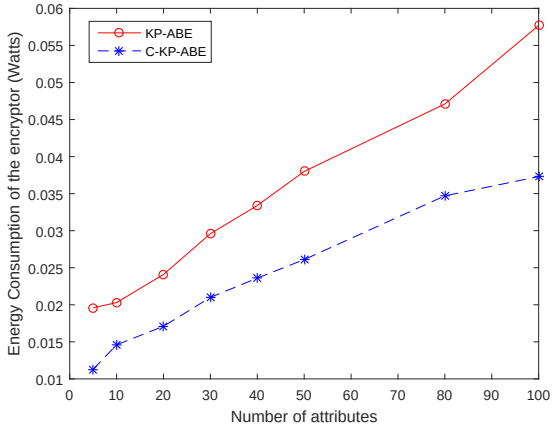



Figure 4.13: Comparison in terms of Energy consumption with respect to number of attributes

**Proposition:** If $D = \prod_{j=1}^{p} D_j$, then $e(C, D)$ can be computed this way:

$$e(C, D) = \prod_{j=1}^{p} e(C, D_j). \tag{4.15}$$

*Proof:*

Let:

$$D = \prod_{j=1}^{p} D_j \tag{4.16}$$

Since each $D_j$ $(j = 1, \cdots, p)$ is from $\mathbb{G}_2$, we can write it this way:

$$\forall j \in \{1, \cdots, p\} : D_j = g^{\alpha_j} \tag{4.17}$$

By replacing in formula 4.16 we get:

$$\begin{aligned} e(C, D) &= e(C, \prod_{j=1}^{p} D_j) \\ &= e(C, \prod_{j=1}^{p} g^{\alpha_j}) \\ &= e(C, g^{\sum_{j=1}^{p} \alpha_j}) \\ &= e(C, g)^{\sum_{j=1}^{p} \alpha_j} \\ &= \prod_{j=1}^{p} e(C, g)^{\alpha_j} \\ &= \prod_{j=1}^{p} e(C, g^{\alpha_j}) \\ &= \prod_{j=1}^{p} e(C, D_j). \end{aligned} \tag{4.18}$$

In order to delegate a pairing operation ($e(C, D)$ for example) to a set of $p$ assistant nodes, a constrained device has only to split the element $D \in \mathbb{G}_2$ into $p$ parts $D_j$ (Where $j \in \{1, \cdots, p\}$) such that the condition in Formula 4.16 is verified.

After that, the constrained device has to send each element $D_j$ to the corresponding assistant node $AN_j$ along with the element $C$. Each assistant node $AN_j$ will compute $e(C, D_j)$ then send it back to the constrained device. Finally, the latter has just to compute the multiplication of all the intermediate results as shown in Formula 4.18.

Usually, $C$ element (the first parameter of the pairing) is part of the ciphertext (See Formulas 3.5 and 3.6 in Section 3.2.3, and Formula 3.11 in Section 3.2.3), and $D$ (the second parameter of the pairing) is a part of the user's secret key. So, the user has to split its secret key parts that are involved in the decryption and then send each part to the corresponding assistant node. After that, it can compute the final result by multiplication.

### 4.7.2 Delegation for CP-ABE decryption

We notice that the decryption primitive of CP-ABE (Section 3.2.3) requires to compute two (2) pairings operations for every leaf node and one additional pairing at the end ($2|Y_{cpabe}| + 1$). It also costs one (1) exponentiation for every non leaf node.

The exponentiation operations of the decryption primitive can be delegated to trusted assistant nodes as we showed it for the encryption primitive (Section 4.3). The pairing operations can also be delegated (Section 4.7.1).

### 4.7.3 Delegation for KP-ABE decryption

In section 3.2.3, we have presented the decryption primitive of KP-ABE. This primitive requires to compute one pairing for every leaf node and one exponentiation for every non leaf node. It is possible to delegate these consuming operations to trusted assistant nodes as we showed it for the encryption primitive in Section 4.4. Pairing operations can be also delegated as we discussed it in Section 4.7.1.

Finally, the resource-constrained device has only to compute multiplications, which is less energy consuming as we have seen it for encryption primitive.

## 4.8 Conclusion

This chapter presents two collaborative approaches for Attribute Based Encryption schemes in the context of the Internet of Things. A resource-constrained device delegates its expensive computational load namely the exponentiation to a set of assistant nodes, on a distributed and cooperative basis. These collaborative approaches leverage the heterogeneity of the Internet of Things to offload the most expensive operations to more powerful trusted assistant.

As we pointed it out in the second chapter, ABE schemes are very complex and induce heavy overhead in term of computation and energy consumption. This drawback is very troublesome when we intend to implement such scheme to secure IoT applications. Indeed, As it is said in the first chapter, IoT may contain highly resource constrained devices that could not support such computation load.

The proposed approaches in this chapter give efficient way to implement two very powerful encryption techniques

Table 4.4: Comparison between CP-ABE and C-CP-ABE ($p = 5$)

| Number of attributes in $Y_{cpabe}$ | Encryption time (s) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | | | a1 | | | d159 | | | f | | |
| | CP-ABE | C-CP-ABE | % | CP-ABE | C-CP-ABE | % | CP-ABE | C-CP-ABE | % | CP-ABE | C-CP-ABE | % |
| 1 | 0.0981 | 0.0167 | 5.86 | 0.7743 | 0.0175 | 44.24 | 0.2058 | 0.0171 | 12.06 | 0.1064 | 0.0192 | 5.54 |
| 5 | 0.3553 | 0.0603 | 5.89 | 2.7265 | 0.0608 | 44.85 | 1.2392 | 0.0580 | 21.37 | 0.1932 | 0.0585 | 3.30 |
| 10 | 0.7209 | 0.0986 | 7.31 | 5.1818 | 0.1064 | 48.69 | 2.5326 | 0.1030 | 24.58 | 0.2975 | 0.1012 | 3.30 |
| 15 | 1.0352 | 0.1456 | 7.11 | 7.5498 | 0.1578 | 47.85 | 4.8398 | 0.1441 | 33.57 | 0.4054 | 0.1472 | 2.75 |
| 20 | 1.3653 | 0.2045 | 6.67 | 10.0306 | 0.1940 | 51.69 | 5.7809 | 0.1886 | 30.64 | 0.5088 | 0.1901 | 2.68 |
| 25 | 1.8326 | 0.2316 | 7.91 | 12.3780 | 0.2502 | 49.47 | 6.8430 | 0.2423 | 28.24 | 0.6201 | 0.2335 | 2.65 |
| 30 | 2.0026 | 0.2870 | 6.98 | 14.7521 | 0.2830 | 52.12 | 9.7338 | 0.2793 | 34.84 | 0.7235 | 0.2787 | 2.60 |
| 35 | 2.4007 | 0.3152 | 7.61 | 17.1951 | 0.3261 | 52.73 | 11.8746 | 0.3616 | 32.84 | 0.8632 | 0.3221 | 2.68 |
| 40 | 2.6879 | 0.4094 | 6.56 | 19.9540 | 0.3836 | 52.01 | 12.3464 | 0.4082 | 30.25 | 0.9464 | 0.3910 | 2.42 |
| 45 | 3.2024 | 0.4479 | 7.15 | 22.3767 | 0.4177 | 53.56 | 13.1266 | 0.4217 | 31.12 | 1.0315 | 0.3983 | 2.59 |
| 50 | 3.3587 | 0.4569 | 7.35 | 25.2817 | 0.4671 | 54.13 | 13.9131 | 0.4591 | 30.31 | 1.1905 | 0.4608 | 2.58 |

Table 4.5: Impact of the number of assistant nodes in C-CP-ABE ($|Y_{cpabe}| = 50$)

| Number of assistant nodes $p$ | encryption time (s) | | | |
|---|---|---|---|---|
| | a | a1 | d159 | f |
| 2 | 0.1525 | 0.1603 | 0.1547 | 0.1545 |
| 5 | 0.4854 | 0.4957 | 0.4763 | 0.4468 |
| 10 | 0.9450 | 0.9675 | 0.9648 | 0.9401 |
| 15 | 1.4443 | 1.6121 | 1.4377 | 1.4521 |
| 20 | 1.9140 | 2.0488 | 1.9412 | 1.9230 |
| 25 | 2.4283 | 2.4695 | 2.4228 | 2.4249 |
| 30 | 2.9694 | 2.9475 | 2.9093 | 2.9181 |
| 35 | 3.4149 | 3.4740 | 3.4093 | 3.3901 |
| 40 | 3.9661 | 3.9450 | 3.9390 | 3.8873 |
| 45 | 4.3675 | 4.5007 | 4.4235 | 4.7648 |
| 50 | 4.8851 | 4.9472 | 4.9236 | 4.8696 |

Table 4.6: Comparison between KP-ABE and C-KP-ABE ($p = 5$)

| Number of attributes in $\gamma$ | Encryption time (s) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | | | a1 | | | d159 | | | f | | |
| | KP-ABE | C-KP-ABE | % | KP-ABE | C-KP-ABE | % | KP-ABE | C-KP-ABE | % | KP-ABE | C-KP-ABE | % |
| 1 | 0.0278 | 0.0085 | 3.27 | 0.2656 | 0.0087 | 30.47 | 0.015 | 0.0087 | 1.72 | 0.06451 | 0.0099 | 6.5 |
| 10 | 0.4582 | 0.0086 | 18.92 | 2.4052 | 0.0088 | 273.35 | 0.0719 | 0.0088 | 8.17 | 0.1146 | 0.0101 | 11.38 |
| 20 | 0.4660 | 0.0087 | 52.82 | 4.7388 | 0.0089 | 533.91 | 0.1821 | 0.0089 | 20.56 | 0.2198 | 0.0101 | 21.81 |
| 30 | 0.1633 | 0.0087 | 53.65 | 7.1067 | 0.0090 | 792.7 | 0.1830 | 0.0089 | 20.64 | 0.2255 | 0.0102 | 22.17 |
| 40 | 0.6182 | 0.0087 | 71.26 | 9.5147 | 0.0091 | 1050.2 | 0.2391 | 0.0089 | 26.93 | 0.2809 | 0.0103 | 27.16 |
| 50 | 0.7765 | 0.0087 | 88.82 | 11.6754 | 0.0092 | 1273 | 0.3002 | 0.009 | 33.36 | 0.3375 | 0.0104 | 32.51 |
| 60 | 0.9377 | 0.0089 | 105.83 | 14.1865 | 0.0093 | 1523.2 | 0.3476 | 0.0092 | 37.97 | 0.3953 | 0.0104 | 38.04 |
| 70 | 1.0796 | 0.0089 | 120.51 | 16.4266 | 0.0093 | 1765.7 | 0.3922 | 0.0092 | 42.69 | 0.4410 | 0.0105 | 42.19 |
| 80 | 1.2451 | 0.0091 | 136.63 | 18.6378 | 0.0094 | 1984.6 | 0.4607 | 0.0092 | 49.96 | 0.5047 | 0.0106 | 47.82 |
| 90 | 1.4239 | 0.0092 | 154.62 | 21.3188 | 0.0102 | 2096.8 | 0.5281 | 0.0094 | 55.99 | 0.5618 | 0.0107 | 52.44 |
| 100 | 1.6155 | 0.0093 | 174.08 | 23.3909 | 0.0095 | 2461.8 | 0.5774 | 0.0095 | 60.86 | 0.6249 | 0.0108 | 57.8 |

Table 4.7: Impact of the number of assistant nodes in C-KP-ABE ($|\gamma| = 100$)

| Number of assistant nodes $p$ | encryption time (s) | | | |
|---|---|---|---|---|
| | a | a1 | d159 | f |
| 2 | 0.002861824 | 0.00304849 | 0.003085261 | 0.003622552 |
| 5 | 0.009277916 | 0.009636145 | 0.009283125 | 0.010444843 |
| 10 | 0.019441771 | 0.02235401 | 0.021232449 | 0.023030574 |
| 15 | 0.037229531 | 0.032086354 | 0.035874844 | 0.04000901 |
| 20 | 0.041470104 | 0.043237761 | 0.048147709 | 0.054405781 |
| 25 | 0.05203901 | 0.058325937 | 0.058128751 | 0.065136927 |
| 30 | 0.062259218 | 0.063950521 | 0.070882344 | 20.076679947 |
| 35 | 0.073235469 | 0.082605521 | 0.08156599 | 0.097693802 |
| 40 | 0.083145364 | 0.085025937 | 0.092088177 | 0.102891823 |
| 45 | 0.093832239 | 0.095871458 | 0.101939531 | 0.118192603 |
| 50 | 0.107791041 | 0.106621302 | 0.113973021 | 0.131379115 |

Table 4.8: Energy consumption for CP-ABE and C-CP-ABE with respect to the number of assistant nodes $p$

| Number of assistant nodes $p$ | C-CP-ABE (watts) | CP-ABE (watts) |
|---|---|---|
| 2 | 0.125 | 0.3138 |
| 3 | 0.15 | 0.3138 |
| 5 | 0.193 | 0.3138 |
| 10 | 0.249 | 0.3138 |
| 15 | 0.299 | 0.3138 |
| 20 | 0.415 | 0.3138 |
| 30 | 0.545 | 0.3138 |
| 40 | 0.668 | 0.3138 |
| 50 | 0.878 | 0.3138 |

Table 4.9: Comparison in terms of Energy consumption

| Number of Attribute in $Y_{cpabe}$ | C-CP-ABE (watts) | CP-ABE (watts) |
|---|---|---|
| 5 | 0.142 | 0.34 |
| 10 | 0.164 | 0.5 |
| 20 | 0.197 | 0.757 |
| 30 | 0.218 | 1.02 |
| 40 | 0.289 | 1.18 |
| 50 | 0.338 | 1.32 |
| 80 | 0.416 | 1.8 |
| 100 | 0.421 | 2.63 |

Table 4.10: Comparison between KP-ABE and C-KP-ABE in terms of Energy consumption

| Number of assistant nodes $p$ | C-KP-ABE (watts) | KP-ABE (watts) |
|---|---|---|
| 2 | 0.0110 | 0.0196 |
| 3 | 0.0113 | 0.0196 |
| 5 | 0.0128 | 0.0196 |
| 10 | 0.0153 | 0.0196 |
| 15 | 0.0169 | 0.0196 |
| 20 | 0.0194 | 0.0196 |
| 30 | 0.0223 | 0.0196 |
| 40 | 0.0264 | 0.0196 |
| 50 | 0.0303 | 0.0196 |

Table 4.11: Comparison in terms of Energy consumption

| Number of Attribute in $\gamma$ | C-KP-ABE (watts) | KP-ABE (watts) |
|---|---|---|
| 5 | 0.0113 | 0.0196 |
| 10 | 0.0146 | 0.0203 |
| 20 | 0.0171 | 0.0241 |
| 30 | 0.0210 | 0.0296 |
| 40 | 0.0236 | 0.0334 |
| 50 | 0.0261 | 0.0380 |
| 80 | 0.0347 | 0.0471 |
| 100 | 0.0373 | 0.0577 |

# Chapter 5

# Attribute Revocation Mechanisms in ABE

## 5.1 Introduction

As we pointed it out in the Section 3.5, the attribute revocation in ABE systems is a very tricky issue, as one attribute could be shared by many users. Hence, making changes in one user's set of attributes may inevitably affect other users sharing same attributes. In other hand, environments like the Internet of Things may contain large number of objects connected together. Therefore, designing scalable attribute/user revocation mechanisms is a very important step before implementing such access control schemes in IoT.

## 5.2 Splitting Time Axis into Time Slots

Let us consider a system where users unpredictably may gain and/or lose one or many attributes in a completely asynchronous and dynamic way. Users attributes validity periods are then unpredictable. We mean by validity period of attribute for a user the duration in which the user possesses the attribute, and it begins from the moment the Attribute Authority (AA) grants the attribute to the user and it ends when the AA revokes it for that user.

In order to optimize and significantly reduce the complexity and the number of exchanged messages, our solution allows the Attribute Authority (AA) to handle simultaneously a lot of changes in users' secret keys. This can be achieved by splitting time into intervals (referred to as *time slots* or simply *slots*) and letting the Attribute Authority handle all the changes that occur in the same interval (slot) at the beginning of the next one.

### 5.2.1 Model and Requirements

#### Architecture Model

The architecture comprises a special entity named *Attribute Authority* (AA). The latter is responsible for managing the universe of the attributes. It also manages users' sets of attributes. The

Attribute Authority generates from each user's set of attributes a secret key which is sent to the corresponding user.

Each user in the system has its own set of attributes that reflect users' privileges in the application. The user uses his secret key to decrypt ciphertexts whose access trees are satisfied by his set of attributes (See Section 3.2.2). The user's set of attributes may evolve through time according to the changes in his function in the system or his context. Thus, the Attribute Authority must update his key by revoking and/or granting some attributes.

#### Security requirements

In our solution, we target the following security requirements:

- **Backward secrecy:**

  A user receiving one or many new attributes, should not be able to use this new secret key in order to access previous encrypted data. The new key is operational only from the moment it is generated.

- **Forward secrecy:**

  A user losing one or many attributes from his set of attributes, should not be able to use the lost attributes to access to current and future ciphertexts encrypted after the key update.

- **Collusion resistance:**

  Collusion resistance is a required property of any ABE system. Even if many users not satisfying the access policy collude, they can obtain no information about the plaintext of the ciphertext.

- **Data Confidentiality:**

  Unauthorized users who do not have the required attributes satisfying the access policy of a ciphertext must be prevented from accessing the plaintext of the data.

#### Assumption

1. Synchronization: We assume that the system is running a synchronization protocol to ensure synchronization between all entities in the system.

#### Time slot based hash function

In this section, we introduce a new hash function that we will use later to construct our schemes. Our hash function takes two parameters: the first parameter is an element from the set of all

attributes managed by the Attribute Authority, the second one is an integer that represents a time slot identifier.

Let $\mathbb{A}$ denotes the set of all attributes used by the Attribute Authority in the application, and $T$ represents the set of all time slots numbers (We have $T \subset \mathbb{N}$). $\mathbb{G}_0$ is a bilinear group of prime order $p$. We define a one-way hash function $H'$ as follows:

$$\begin{array}{rccc} H': & \mathbb{A} \times T & \longrightarrow & \mathbb{G}_0 \\ & (att, i) & \longmapsto & H'(att, i) \end{array}$$

We suppose that the probability of collision existence in the one-way hash function $H'$ is infinitely small. We mean by collision the existence of two different couples $(att_i, k), (att_j, l) \in \mathbb{A} \times \mathbb{N}$ (with: $(att_i, k) \neq (att_j, l)$), such that $H'(att_i, k) = H'(att_j, l)$. This assumption is described in the following formula:

$$\forall att_i, att_j \in \mathbb{A}, \forall k, l \in T : (att_i, k) \neq (att_j, l) \Rightarrow P\left(H'(att_i, k) = H'(att_j, l)\right) \approx 0 \quad (5.1)$$

We provide in Appendix C 8.3 a simple construction of our hash function.

### 5.2.2 Batch Based CP-ABE

In this section, we present our solution named Batch-Based CP-ABE (BB-CP-ABE). We first present our motivations and the targeted application cases, then, we introduce the basic concept of our approach. And finally, we detail the primitives of our scheme.

#### Motivation and Applications Scenarios

We target applications that don't have hard time constraints. The revocation and granting date may be flexible and be postponed to a later date.

For example, let us consider the information security management system of a hospital. In this case, attributes could be administrative grades (Director, Department Chief, Secretary, Employee, ... etc.), Departments (Cardiology Department, Neurology Department, Emergency Department, ... etc.), functional grades (Nurse, Doctor, Trainee, ... etc.). For example, a trainee who has finished his/her internship in a hospital must see her/his secret key revoked, more precisely, the system should revoke her/his "Trainee" attribute. Thus, he/she can not use her/his secret key part related to the attribute "Trainee" to decrypt ciphertexts. Similarly, a nurse who has moved from cardiology department to emergency one must lose her abilities to decrypt ciphertexts destined

to cardiology department employee, this is resulting in revoking her "Cardiology Department" attribute.

### Concept

Here we describe the basic idea of our solution to implement attribute revocation mechanism in CP-ABE scheme.

We introduce, first, two definitions which are necessary to explain the solution.

- **Definition 1**:

  *Real Attribute Validity Period (RAVP)* is the validity period of an attribute for a particular user which is imposed by the application. For example, a trainee starting his internship from April 1st, 2016 to September 30th, 2016, the real attribute validity period is exactly the period from April 1st, 2016 to September 30th, 2016 (183 days).

- **Definition 2**:

  *Delivery Attribute Validity Period (DAVP)* is the validity period that corresponds to the union of all time slots corresponding to the elements of secret key delivered by the Attribute Authority. This period is usually different from the real attribute validity period because of the lag introduced by our scheme. This period highly depends on the time slot duration chosen in the application.

Our first approach is a *batch-based* method, which means that time axis is split into intervals of the same duration called *time slots*, and policy access changes (granting and/or revoking access) occur only between two successive time slots.

To implement attribute revocation in CP-ABE, our solution is not based on renaming attributes or using access tree to include a policy that considers expiration time for an attribute as proposed in [11] and [56]. In our approach, the Attribute Authority send only the necessary attribute key parts every time slot to allow an entity to refresh its secret key. This technique reduces the overhead and the complexity of the solution comparing to the existing ones.

### Primitives

Let $\mathbb{G}_0$ be a bilinear group of prime order $p$, and let $g$ be a generator of $\mathbb{G}_0$. In addition, let $e$ : denote the bilinear map.

**Setup.** It chooses a bilinear group $\mathbb{G}_0$ of prime order $p$ with generator $g$. Next it will choose two random exponents $\alpha, \beta \in Z_p$. The public key is published as:

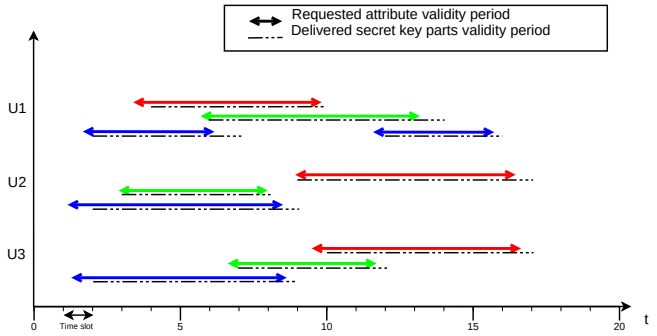



Figure 5.1: Example

| Users | Attributes | Number of time slots | Time slots list |
|---|---|---|---|
| U1 | Att 1 | 5, 6, 7, 8, 9, 10 | 6 |
| | Att 2 | 7, 8, 9, 10, 11, 12, 13, 14 | 8 |
| | Att 3 | 3, 4, 5, 6, 7, 12, 13, 14, 15, 16 | 10 |
| U2 | Att 1 | 10, 11, 12, 13, 14, 15, 16, 17 | 8 |
| | Att 2 | 4, 5, 6, 7, 8 | 5 |
| | Att 3 | 3, 4, 5, 6, 7, 8, 9 | 7 |
| U3 | Att 1 | 11, 12, 13, 14, 15, 16, 17 | 7 |
| | Att 2 | 8, 9, 10, 11, 12 | 5 |
| | Att 3 | 3, 4, 5, 6, 7, 8, 9 | 7 |

Table 5.1: Example

$$PK = \left(\mathbb{G}_0, g, h = g^{\beta}, f = g^{1/\beta}, e\left(g, g\right)^{\alpha}\right) \tag{5.2}$$

and the master key is:

$$MK = (\beta, g^{\alpha}) \tag{5.3}$$

Note that $f$ is used only for delegation, so we can omit it here as we do not talk about a delegation primitive. For more information we invite the reader to see [11].

**KeyGen***(MK, S)*.

This KeyGen primitive takes as input the master key $MK$ and a set $S$ which contains a set of attributes and all the corresponding time slot numbers of their validity period. We can write the set $S$ as:

$$S = \left\{\left(a, T^a_{begin}, T^a_{end}\right), \text{for all attribute } a\right\} \tag{5.4}$$

The Key generation algorithm chooses a random $r \in Z_p$, and then random $r_j \in Z_p$ for each attribute $j \in A$. Then it computes the key as

$$SK = \Bigg( D = g^{(\alpha+r)/\beta}, \forall j \in A, \forall k \in [\![T^j_{begin}, T^j_{end}]\!] : \\ D_{j,k} = g^r H(j,k)^{r_j}, D'_j = g^{r_j} \Bigg) \quad (5.5)$$

Note here that the parameter $D_{j,k}$ is related to the attribute $j$ for the time slot $k$.

**Encrypt***(PK, M, $\gamma$, T)*.

The encryption primitive encrypts a message $M$ under the tree access $\gamma$ and the time slot $T$. The algorithm first chooses a polynomial $q_x$ for each node $x$ (including the leaves) in the access tree $\gamma$. These polynomials are chosen in the following way in a top-down manner, starting from the root $R$. For each node $x$ in the tree, set the degree $d_x$ of the polynomial $q_x$ to be one less than the threshold value $k_x$ of that node, that is, $d_x = k_x - 1$.

Starting with the root node $R$ the algorithm chooses a random $s \in \mathbb{Z}_p$ and sets $q_R(0) = s$. Then, it chooses $d_R$ other points of the polynomial $q_R$ randomly to define it completely. For any other node $x$, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses $d_x$ other points randomly to completely define $q_x$.

Let $Y$ be the set of leaf nodes in $\gamma$. The ciphertext is then constructed by giving the tree access structure $\gamma$, the decryption time slot $T$ and computing:

$$CT = \Bigg( \gamma, T, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \\ \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y), T)^{q_y(0)} \Bigg) \quad (5.6)$$

Only users that satisfy the access tree $\gamma$ during the time slot $T$ can decrypt the ciphertext $CT$. We notice here that a user encrypting a message during the time slot $T_1$ can specify a different time slot $T_2$ for decryption.

**Decrypt***(CT, $SK_T$)*.

The decryption primitive takes the ciphertext $CT$ and a secret key $SK_T$, it is quite similar to the first form proposed in [11] with the unique difference in using our one-way hash function $H$ defined in Section 5.2.1 instead of a standard one.

We first define a recursive function $DecryptNode(CT, SK_T, x)$ that takes as input a ciphertext $CT = \left(A, T, \tilde{C}, C, \forall y \in Y : C_y, C'_y\right)$, a secret key $SK_T = \left(D, \forall j \in S_T : D_{j,T}, D'_j\right)$, which is

associated with a set $S_T$ of valid attributes at the time slot $T$, and a node $x$ from the access tree $A$.

If the node $x$ is a leaf node then we let $i = att(x)$ and define $DecryptNode$ as follows: If $i \in S_T$, then

$$\begin{aligned} DecryptNode(CT, SK_T, x) &= \frac{e(D_{i,T}, C_x)}{e(D'_i, C'x)} \\ &= \frac{e\left(g^r \cdot H(i,T)^{r_i}, g^{q_x(0)}\right)}{e\left(g^{r_i}, H(i,T)^{q_x(0)}\right)} \\ &= e(g,g)^{rq_x(0)}. \end{aligned}$$

If $i \notin S_T$, then $DecryptNode(CT, SK_T, x) = \perp$.

Now, we consider the recursive case when $x$ is a non-leaf node. The algorithm $DecryptNode(CT, SK_T, x)$ then proceeds as follows: For all nodes $z$ that are children of $x$, it calls $DecryptNode(CT, SK_T, z)$ and stores the output as $F_z$. Let $S_x$ be an arbitrary $k_x$-sized set of child nodes $z$ such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$.

Otherwise, we compute

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}; \text{ Where: } i = index(z), S'_x = \{index(z) : z \in S_x\} \\ &= \prod_{z \in S_x} \left(e(g,g)^{r \cdot q_z(0)}\right)^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} \left(e(g,g)^{r \cdot q_{parent(z)}(index(z))}\right)^{\Delta_{i,S'_x}(0)} \text{ (by construction)} \\ &= \prod_{z \in S_x} e(g,g)^{r \cdot q_x(i) \cdot \Delta_{i,S'_x}(0)} \\ &= e(g,g)^{r \cdot q_x(0)} \text{ (Using polynomial interpolation)} \end{aligned}$$

and return the result.

After defining our function DecryptNode, we can now define the decryption algorithm. The algorithm begins by calling the function on the root node $R$ of the tree $A$. If the tree is satisfied by $S_T$ we set $A = DecryptNode(CT, SK_T, r) = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$. The algorithm now decrypts by computing

$$\begin{aligned}\tilde{C}/\left(e\left(C,D\right)/A\right) &= \tilde{C}/\left(e\left(h^s, g^{(\alpha+r)/\beta}\right)/e\left(g,g\right)^{rs}\right)\\ &= Me\left(g,g\right)^{\alpha s}/\left(e\left(g,g\right)^{s(\alpha+r)}/e\left(g,g\right)^{rs}\right)\\ &= M.\end{aligned}$$

### 5.2.3 Variable Time Slots Durations

In this Section, we present our second solution to implement efficiently attribute revocation with CP-ABE. First, we motivate the problem. Then, we present the concept of the solution. Finally, we detail the primitives.

#### Motivations

Our motivation to propose this solution lies in the limitations of the previous solution presented in Section 5.2.2. Indeed, Batch-Based CP-ABE induces a delay between the real attribute validity period and the delivery attribute validity period. This drawback may prevent the previous solution to be applicable in some application where hard time constraint are imposed.

In this solution, we aim to eliminate this lag between the two periods while respecting security requirements (See Section 5.2.1). The solution (that we call Instantaneous Batch-Based CP-ABE) must ensure an immediate revocation without any delay between real attribute validity period and the delivery attribute validity period.

This solution targets applications where the real validity period of all attributes and for all users are know by the Attribute Authority. The latter will generate users' secret keys based on their attributes validity periods.

#### Concept

As we pointed it out in the previous section, the Attribute Authority must know beforehand every attribute validity period for all users. The targeted applications are, for example, institutions system management where permanent users' roles do not evolve rapidly. For instance, in e-health, where doctors, nurses, etc. hold a set of attributes reflecting their roles and functions in the system. These attributes have known validity periods and start from known dates.

In our solution, the Attribute Authority begins by collecting all attributes validity periods of users. Then, it determines for each attribute, separately, the series of time slots with variable durations as shown in Figure 5.2. Then, the Attribute Authority assigns an identifier to each time slot and determines the number of secret key parts to generate and send to each user according to their attribute validity periods.

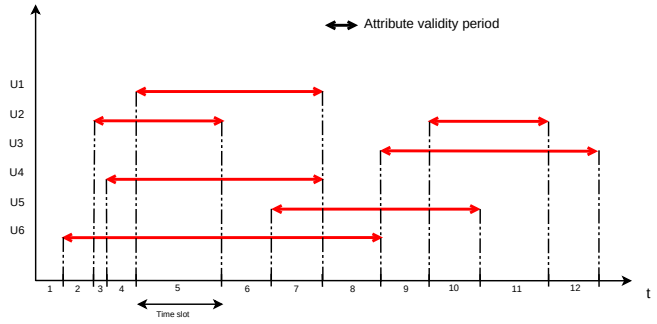



Figure 5.2: Example of creating time slots with variable durations

In Figure 5.2, axis of ordinates contains different users ($U1, U2, ..., U6$), and axis of abscissa represents time which is split into different time slots with different durations. Vertical projections of time events [1] upon axis abscissa are shown with dotted lines. These projections will determine beginnings and ends of time slots.

For example, user $U1$ has a validity period that extends over three time slots (5, 6 and 7). Table 5.2 shows the number of time slots and their assignment by the Attribute Authority to each user.

Table 5.2: Example.

| | Number of time slots | Corresponding time slots |
|---|---|---|
| U1 | 3 | 5,6,7 |
| U2 | 5 | 3, 4, 5, 10, 11 |
| U3 | 4 | 9, 10, 11, 12 |
| U4 | 4 | 4, 5, 6, 7 |
| U5 | 4 | 7, 8, 9, 10 |
| U6 | 7 | 2, 3, 4, 5, 6, 7, 8 |

When an attribute related event occurs (Attribute validity period starts or ends), the Attribute Authority increments the time slot identifier $T_{att}$ related to that attribute and informs all entities in the system. For each attribute, our solution generates as much secret key parts (SKP) as time slots in the validity period of that attribute. The user shifts easily to the new secret key associated to the current time slot without being able to generate secret keys for time slots outside the attribute validity scope for the user.

This solution uses also the one way hash function introduced in Section 5.2.1 to generate the secret key part for every time slots.

### Primitives

Let $\mathbb{G}_0$ be a bilinear group of prime order $p$, and let $g$ be a generator of $\mathbb{G}_0$. In addition, let $e$ denote the bilinear map.

[1] We mean by event any attribute validity period beginning or ending.

There are four cryptographic primitives:

**Setup**. The setup algorithm is run by the Attribute Authority at the bootstrap phase. It takes no input other than the implicit security parameter. It outputs the public parameters $PK$ which is shared with all the entities of the system and a master key $MK$ kept secret.

The algorithm operates as follows. It chooses a bilinear group $\mathbb{G}_0$ of prime order $p$ with generator $g$. Next it will choose two random exponents $\alpha, \beta \in Z_p$. The public key is published as:

$$PK = \left(\mathbb{G}_0, g, h = g^{\beta}, f = g^{1/\beta}, e\left(g, g\right)^{\alpha}\right) \tag{5.7}$$

and the master key is:

$$MK = (\beta, g^{\alpha}) \tag{5.8}$$

**KeyGen***(MK, S)*. The KeyGen primitive is run by the Attribute Authority for each user joining the system. It takes as input the master key $MK$ and a set of couples $S$. Each element of the set $S$ contains two parts: the first one is an attribute $att \in A$, and the second one is a list of time slots numbers $TSL_{att}$ defining the validity period of the attribute $att$.

We can write the set $S$ as:

$$S = \left\{\left(att, TSL_{att}\right), \forall att \in A\right\} \tag{5.9}$$

The Key generation algorithm begins by choosing a random $r \in Z_p$, and then a random $r_j \in Z_p$ for each attribute $j \in A$. Then, it computes the key as follows:

$$SK = \left(D = g^{(\alpha+r)/\beta}, \forall j \in A, \forall k \in TSL_j : D_{j,k} = g^{r} \cdot H\left(j, k\right)^{r_j}, D'_j = g^{r_j}\right) \tag{5.10}$$

Note here that the parameter $D_{j,k}$ is related to the attribute $j$ for the time slot number $k$.

In formula 5.10, $SK$ represents a user *global* secret key throughout the lifetime of the system; it contains all the subkeys that are used to decrypt ciphertexts. At a specific time, the user uses one of these subkeys to decrypt data. The subkeys are extracted from the global secret key $SK$ by keeping only the elements related to the current time slot number for each attribute in $A$. Let $TSL$ be a list of time slots identifiers representing the current time slots identifiers of all attributes in $A$. The subkey related to $TSL$ is noted $SK_{TSL}$ and is computed as following:

$$SK_{TSL} = \left(D, \forall j \in A : D_{j,TSL(j)}, D'_j\right) \tag{5.11}$$

The writing $TSL(j)$ means the element of $TSL$ which is related to the attribute $j$. It represents the current time slot identifier of the attribute $j$.

**Encrypt***(PK, M, $\gamma$, $TSL$).* The encryption algorithm takes as input the public parameters $PK$, a message $M$, an access structure $\gamma$ over the universe of attributes and a time slots list $TSL$ containing a list of current time slots numbers associated with the attributes of the access structure leaf nodes. The algorithm will encrypt $M$ and produce a ciphertext $CT$ such that only a user that possesses a set of attributes, during their corresponding time slots in $TSL$, that satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains $\gamma$ and $TSL$.

The encryption primitive operates in the same manner as the standard version defined in [11] except in using our hash function defined in 5.2.1. Each attribute in leaf nodes of the access tree $\gamma$ has its corresponding time slot number in $TSL$. The algorithm first chooses a polynomial $q_x$ for each node $x$ in the access tree $\gamma$. These polynomials are chosen in top-down manner, starting from the root node $R$ down to leaf nodes. For each node $x$ in the tree, the degree of the polynomial $q_x$ is set to be one less than the threshold value $k_x$ of that node: $d_x = k_x - 1$.

The algorithm chooses a random $s \in Z_p$ and sets $q_R(0) = s$. Then, chooses $d_R$ other points of the polynomial $q_R$ randomly to define it comletely. For any other node $x$, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses $d_x$ other points randomly to define $q_x$.

Let, $Y$ be the set of leaf nodes in $\gamma$. $Y$ and $TSL$ have the same size, and every element $y \in Y$ has its corresponding element $TSL(y) \in TSL$. The ciphertext is the constracted by giving the access tree $\gamma$, the current time slot number for each element in $Y$, and computing:

$$CT = \Big(\gamma, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \forall y \in Y : TSL(y), C_y = g^{q_y(0)}, C'_y = H(att(y), TSL(y))^{q_y(0)}\Big) \quad (5.12)$$

**Decrypt***(CT, $SK_{TSL}$).*

The decryption algorithm takes as input a ciphertext $CT$, which contains an access policy $\gamma$, and a private key $SK_{TSL}$ constructed from a list $A$ of attributes associated to the time slots list $TSL$. The time slots list $TSL$ used here is the same as the one used for constructing the ciphertext $CT$. If the set $A$ associated to a time slots list $TSL$ of attributes satisfies the access structure $\gamma$, then the algorithm will be able to decrypt the ciphertext and return a message $M$.

The decryption primitive is pretty similar to the one defined in [11] except in using our hash function defined in Section 5.2.1. We first define $DecryptNode(CT, SK_{TSL}, x)$ which is a recursive function. It takes a ciphertext $CT = \Big(\gamma, \tilde{C}, C, \forall y \in Y : TSL(y), C_y, C'_y\Big)$, a private key $SK_{TSL} = \Big(D, \forall j \in A : D_{j,TSL(j)}, D'_j\Big)$ which is associated with a set $A$ of attributes, and a node $x$ from $\gamma$.

*Case 1:* The node $x$ is a leaf node, then we let $i = att(x)$. If $i \in A$, then

$$
\begin{aligned}
DecryptNode\left(CT, SK_{TSL}, x\right) &= \frac{e\left(D_{i,T}, C_x\right)}{e\left(D_i', C'x\right)} \\
&= \frac{e\left(g^r \cdot H\left(i, TSL\left(i\right)\right)^{r_i}, g^{q_x(0)}\right)}{e\left(g^{r_i}, H\left(i, TSL\left(i\right)\right)^{q_x(0)}\right)} \\
&= e\left(g, g\right)^{rq_x(0)}.
\end{aligned}
$$

and if $i \notin A$, then $DecryptNode\left(CT, SK_{TSL}, x\right) = \perp$.

*Case 2:* The node $x$ is a not leaf node.

The algorithm proceeds as follows: For all nodes $z$ that are children of $x$, it calls $DecryptNode\left(CT, SK_{TSL}, z\right)$ and stores the output as $F_z$.

Otherwise, we compute

$$
\begin{aligned}
F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S_x'}(0)}; \\
&= \prod_{z \in S_x} \left(e\left(g, g\right)^{r \cdot q_z(0)}\right)^{\Delta_{i,S_x'}(0)} \\
&= \prod_{z \in S_x} \left(e\left(g, g\right)^{r \cdot q_{parent(z)}(index(z))}\right)^{\Delta_{i,S_x'}(0)} \\
&= \prod_{z \in S_x} e\left(g, g\right)^{r \cdot q_x(i) \cdot \Delta_{i,S_x'}(0)} \\
&= e\left(g, g\right)^{r \cdot q_x(0)} \text{ (Using polynomial interpolation)}
\end{aligned}
$$

Where $i = index\left(z\right), S_x' = \{index\left(z\right) : z \in S_x\}$.

We recall that $\Delta_{i,S}\left(x\right)$ is the Lagrange coefficient defined as follows:

$$
\Delta_{i,S}\left(x\right) = \prod_{j \in S, j \neq i} \left(x - j\right) / \left(i - j\right).
$$

where $i$ be an element in $\mathbb{Z}_p$, and $S$ a set of elements in $\mathbb{Z}_p$.

### 5.2.4 Performance Analysis

#### Simulation Model

For the sake of simplicity and without loss of generality, we consider a group of users which ask gaining the access right to one attribute. Results can be easily extrapolated when considering

multiple independent attributes. We modeled users' requests, which represent starting dates of attribute validity periods, by Poisson process with parameter $\lambda$. The attribute validity periods durations for all users follow exponential distribution with parameter $\mu$. The following simulations are made considering a system with one thousand (1000) entities.

We are interested in evaluating the overhead in terms of generated secret key parts (SKP) for an attribute[2]. This metric is very important since it determines the size of generated secret key sent by the Attribute Authority to system entities. $SKP$ reflects closely system performances. Indeed, the less is the number of secret key parts $SKP$, better is the solution.

**Average Number of time slots (ANT)**

The figure 5.3 shows the impact of the time slot duration on the system performance i.e. the average number of time slots per user. The figure shows also the impact of the parameter $\mu$ on the average number time slots per user. From this figure and figure 5.4, we can deduce empirically that the average number of time slots $ANT$ is inversely proportional to the chosen time slot duration $\Delta t$ (See Appendix A 8.1):

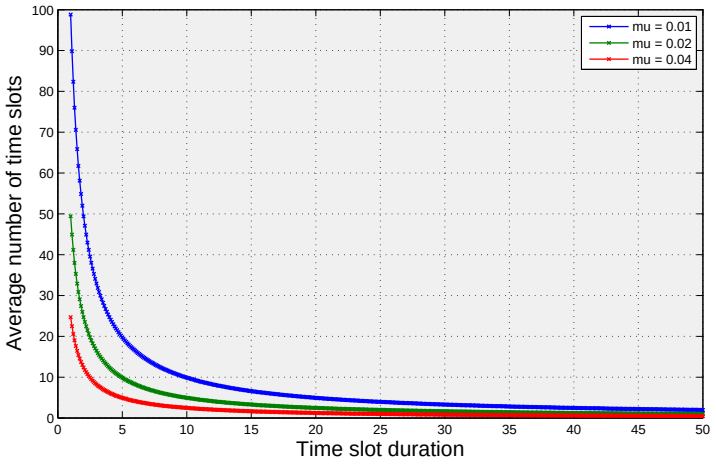



Figure 5.3: Average number of time slots

$$ANT \approx \frac{1}{\mu \Delta t} \tag{5.13}$$

[2]We mean by secret key part the element $D_{j,k}$ in the secret key $SK$ (see Section 5.2.2 and Section 5.2.3)

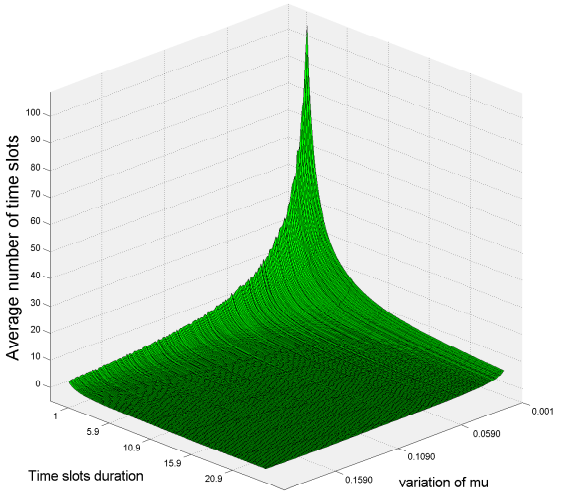



Figure 5.4: Average number of time slots

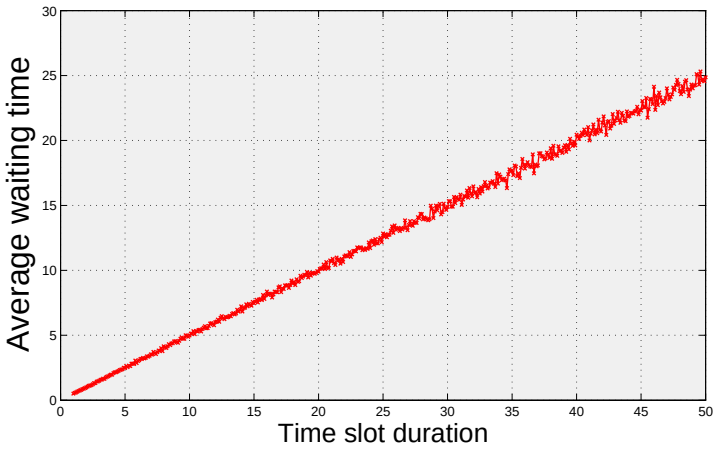



Figure 5.5: Average waiting time

**Average Waiting Time (AWT)**

We are also interested on the average waiting time $AWT$, figure 5.5 shows the result of a experiment simulating the variation of the overall average waiting time of all users depending on the the time slot duration. We set $\lambda = 0.01$. We note that the average waiting time $AWT$ is independent of the Poisson process parameter $\lambda$, on the other side it increases linearly with the time slot duration $\Delta t$, we can deduce empirically (See Appendix B 8.2):

$$AWT \approx 1/2 \cdot \Delta t \tag{5.14}$$

**ANT-vs-AWT**

We notice here that we are trying to minimize both of the Average Waiting Time $AWT$ and the Average Number of Time slots per user $ANT$ which are two conflicting objectives i.e. minimizing $AWT$ leads to maximizing $ANT$ and vice versa. In real life applications we choose a tradeoff between performance and delay.

From formulas 5.14 and 5.13, we can deduce that:

$$ANT \approx \frac{1}{2\mu AWT} \tag{5.15}$$

and this is shown in figure 5.6.

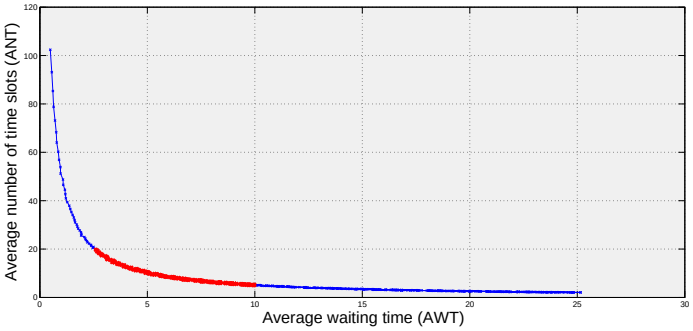



Figure 5.6: AWT vs ANT

In real applications we define some conditions and constraints related to the system requirements such as:

$$\begin{cases} AWT < \alpha/\mu \\ ANT < \beta/\mu \end{cases}$$

$\alpha$ and $\beta$ are to be determined according to the application requirements. In Figure 5.6, we show in red color the set of solutions that satisfy these conditions where $\alpha = 1/10$ and $\beta = 1/5$.

**Number of SKP**

Figure 5.7 shows the variation of the secret key parts $SKP$ needed to be sent with respect to Poisson process parameter $\lambda$. The value fixed for $\mu$ is $0.1$. The number of generated secret key parts $SKP$ increases almost linearly with the $\lambda$. our solution shows better results in applications cases where $\lambda$ is small.

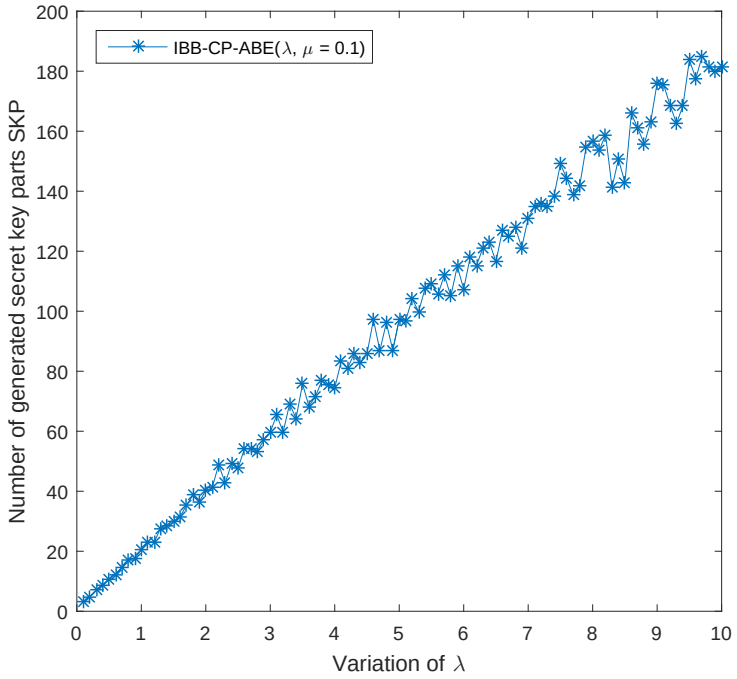



Figure 5.7: Number of generated secret key parts SKP with respect to $\lambda$

Figure 5.8 shows the variation of the secret key parts needed to be sent with respect to the exponential distribution parameter $\mu$. We fixed $\lambda = 10$.

According to the two figures 5.7 and 5.8 we approximate $SKP(\lambda, \mu)$ by the following formulas:

$$SKP(\lambda, \mu) \approx 2 * \lambda/\mu + 1 \tag{5.16}$$

Figure 5.9 shows the variation of the number of generated secret key part $SKP$ with respect to both $\lambda$ and $\mu$.

### Comparisons

In this section we carry out a comparison of our solutions. Recall that in BB-CP-ABE, all time slots have the same duration $\Delta t$ and if attribute validity period beginning and end occur during a time slot, they are delayed until the next time slot begins.

We considered a system with one thousand (1000) users, where attribute starting dates follow Poisson process with parameter $\lambda$ and attribute validity periods have an exponential distribution with parameter $\mu$.

We choose three values for time slot duration $\Delta t$ of Batch-Based CP-ABE solution: $\Delta t1 = 1$, $\Delta t2 = 2$ and $\Delta t3 = 4$. These three cases correspond to three possible configurations of BB-CP-

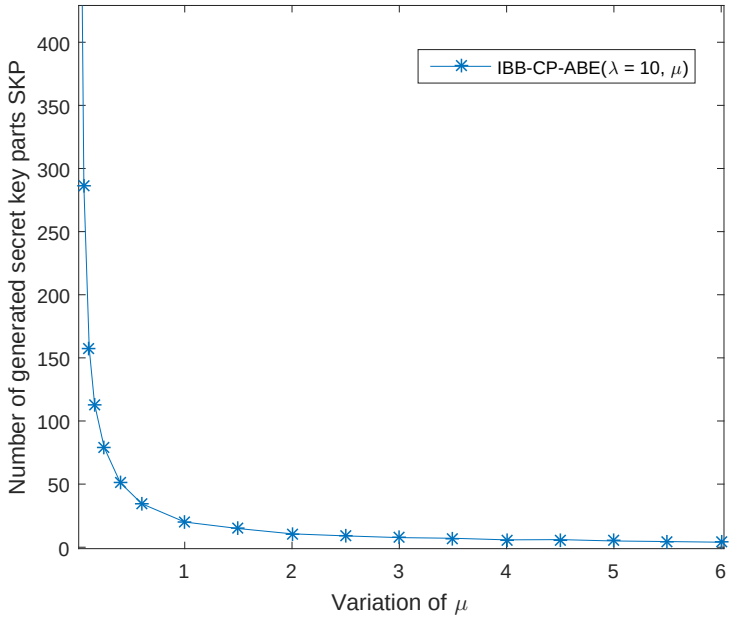



Figure 5.8: Number of generated secret key parts SKP with respect to $\mu$

ABE.

In the first simulation, we set $\mu = 0.1$ and we computed the variation of the numbers of generated secret key parts in the four cases with respect to $\lambda$. Simulation results are shown in figure 5.10.

In the case of IBB-CP-ABE, the number of elements to be sent increases linearly with respect to $\lambda$, this can be explained by the fact that the higher is $\lambda$, the less the time between two incoming users (attribute validity periods); therefore, the more frequent are overlaps between attribute validity periods. The curves in the three cases of BB-CP-ABE are almost constant, this is because the number of elements to be sent does not depend on $\lambda$, it depends only on $\mu$ and time slot duration. For small values of $\lambda$, IBB-CP-ABE shows better performance than BB-CP-ABE. It is important to remember here that IBB-CP-ABE does not induce any delay, The requested validity period is the same as the delivered validity period. but in BB-CP-ABE and according to the time slot duration we have an average delay equals to 1/2, 1, 2 respectively in the case of $\Delta t1$, $\Delta t2$ and $\Delta t3$.

For high values of $\lambda$, the BB-CP-ABE solution overcomes IBB-CP-ABE, this is because BB-CP-ABE could adjust many different user's requested validity periods to the same list of time slots. In other words, The BB-CP-ABE delays managing many attribute revocations to the beginning of the next time slot and treat them all at a time. This is the reason why BB-CP-ABE outperforms IBB-CP-ABE for high values of $\lambda$. However, this particularity of BB-CP-ABE produces a lag between the requested validity period and the delivered validity period. This delay (lag) is undesirable or even inapplicable in most cases for applications that require precision.

In the second simulation, we set $\lambda = 0.1$ and we computed the variation of the numbers of generated secret key parts in the four cases with respect to $\mu$. Simulation results are shown in figure

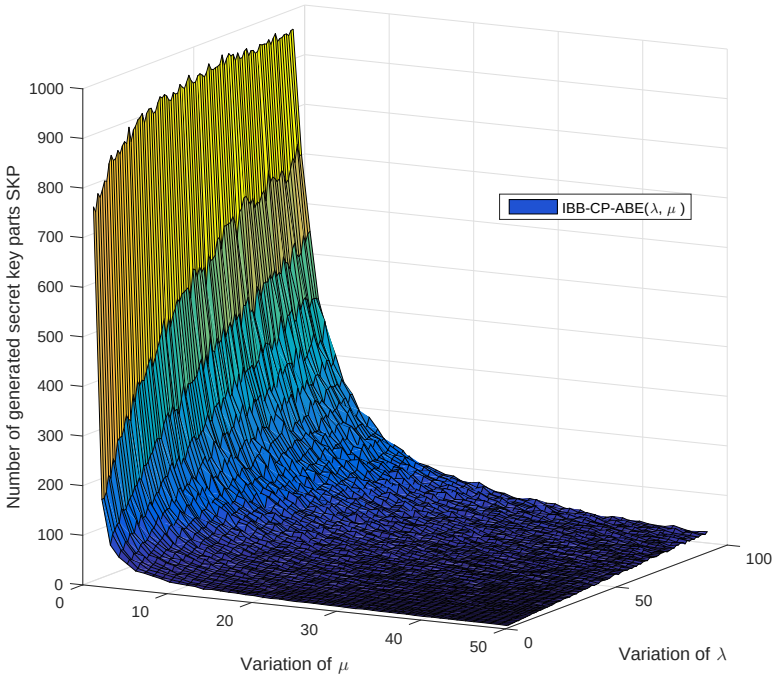



Figure 5.9: Number of generated secret key parts SKP with respect to both $\lambda$ and $\mu$

5.11.

The four curves are inversely proportional to $\mu$. For small values of $\mu$, IBB-CP-ABE shows with dark blue ink gives best results than the three others. But with larger values of $\mu$, IBB-CP-ABE becomes less efficient than the three others. We also recall here that IBB-CP-ABE contrary to others does not induce any delay.

## 5.3 Instantaneous Proxy revocation

Previous attribute revocation solutions require either the a priori knowledge of attributes revocation schedule or postponing the effective revocation of attributes and associated privileges. However, some applications require the immediate revocation of attributes and associated privileges without presenting a schedule of those events. In this section, we present a new solution that fulfill such scenarios and requirements.

### 5.3.1 Network model

We assume the existence of a semi-trusted proxy that assists entities in the decryption process. We mean by semi-trusted that the proxy is curious but honest, it will honestly execute the tasks assigned to it, however, it would like to learn information of encrypted data. The proxy receives

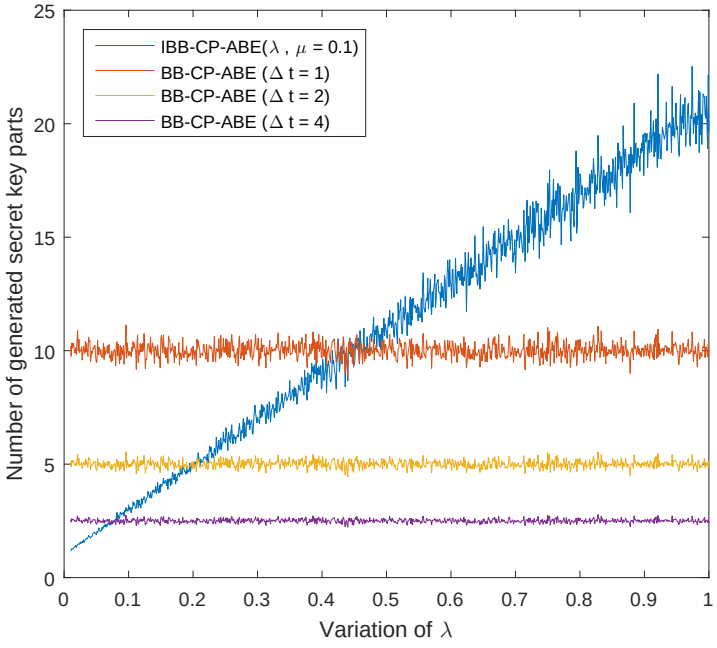



Figure 5.10: Comparison with respect to $\lambda$

parts of entities' secret keys during the key generation phase. An entity called *Attribute Authority* manages users' attributes and creates users' secret keys. This entity has also the role of holding the public parameters and is responsible for the revocation mechanism. All other entities are considered users of the system.

We assume that after the initialization phase, each user $i$ in the system shares a symmetric key $K_{p,i}$ with the proxy. We assume also that the proxy has received a Proxy Secret Key $PrSK$ where the corresponding Proxy Public Key $PrPK$ is shared with all other users.

Figure 5.12 shows the global architecture of our solution illustrating the different involved parties. It shows also the exchanges between these entities. The *Attribute Authority* is responsible for generating the *Public Key* $PK$ and sharing it with the *Data Owners (DO)* (**Step 1**). It also generates to each user a *Secret Key* $SK$ based on the user's attributes set (**Step 2**). The secret key $SK$ is divided into two parts $SK^{(1)}$ and $SK^{(2)}$. The first part ($SK^{(1)}$) is sent to the concerned user, and the other part ($SK^{(2)}$) to the proxy.

Meanwhile, the data owner is able to encrypt his sensitive data and stores them in a remote server (**Step 3**). We recall that, the storage server is not charged of ensuring access control to data; but rather, it is cryptographically implemented by CP-ABE.

The user gets the cipher-text from the storage server (**Step 4**), he solicits the assistance of the proxy to decrypt it (**Step 5**).

The key update process is similar to the key generation (**Step 6**).

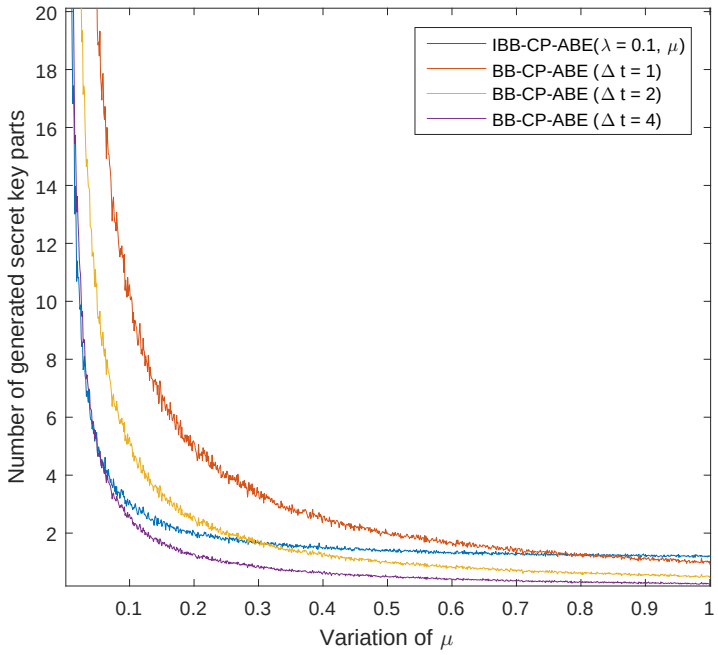



Figure 5.11: Comparison with respect to $\mu$

### 5.3.2 Assumptions

Let $e$ be a non-degenerate bilinear map (See Section 3.2.2). Our scheme is based on two assumptions about pairing inversions:

- The Fixed Argument Pairing Inversion 1 (FAPI-1) [32]: Given $D_1 \in \mathbb{G}_1$ and $z \in \mathbb{G}_T$, compute $D_2 \in \mathbb{G}_2$ such that $e(D_1, D_2) = z$.
- The Fixed Argument Pairing Inversion 2 (FAPI-2) [32]: Given $D_2 \in \mathbb{G}_2$ and $z \in \mathbb{G}_T$, compute $D_1 \in \mathbb{G}_1$ such that $e(D_1, D_2) = z$.

We have also set some assumptions on the different entities in the network model:

1. Each entity $E_i$ of the system shares a symmetric key with the proxy $k_{p,i}$.
2. The proxy possesses a unique Proxy Secret Key $PrSK$ and its corresponding Proxy Public Key $PrPK$ is published and is known by all system entities. $PrSK$ and $PrPK$ are asymmetric keys.
3. The proxy is semi-trusted (honest but curious): It executes its function honestly without disclose any users' information to other parties. However, it could be curious to get as much as possible of information about users' data, or even try to use secret elements of users to access protected data.

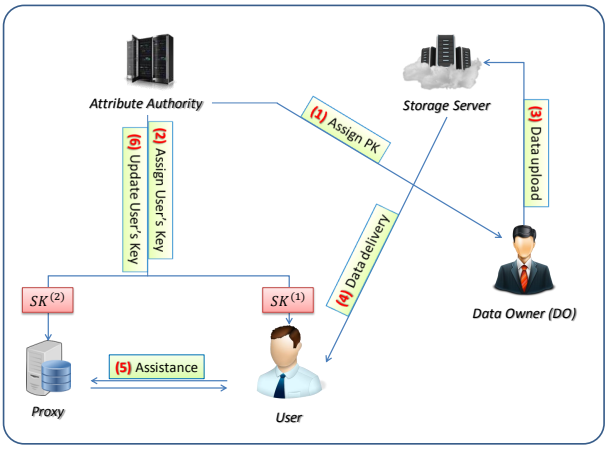



Figure 5.12: Architecture of the solution

### 5.3.3 Solution 1: Backward and Forward Accessibility

#### Motivations and Application cases

In this variant, we aim to tackle the attribute revocation issue of CP-ABE in applications where a user has only access to data that are encrypted with a policy satisfied by her/his current attributes list. In this kind of applications, the encryption date is not significant when decrypting the cipher-text. The only thing that matters is the cipher-text's access policy and the current users' attributes list. Which means that, a user gaining some new attributes has the access right to all data encrypted with an access policy satisfied with the new user's attributes list even if the data was encrypted before updating the user's attributes list. Similarly, a user who loses some attribute will not have the access right to data whose access policy is not satisfied by the new user's attributes list.

**- Medical files management system:** An example of real life applications that follows these security requirements is the medical files management system. Each patient has her/his own medical file that lists all information about her/his health (Medical history, prescribed medication, ... ). These information must be kept secret in order to protect user's privacy. When a doctor has to examine this patient, she/he needs patient's medical file, so she/he must be allowed to access to the whole information in it even if the doctor examines this patient for the first time.

In order to preserve the patient's privacy, when the she/he changes her/his treating doctor, all her/his previous doctors must be prevented from accessing the new updates of her/his medical file. In other words, all the information that would be added by the current doctor must be kept secret from all previous doctors that have ever treated the patient.

**- Meetings management system:** The system that manages projects meetings must ensure that only members working on the project have the access to the relating information. The access right to project information could be materialized by owning the "*project partner*" attribute. When an employee becomes a partner of a project, the system should authorize him to access all previous information about the project. So as he can pursue the project progression easily.

When a partner employee leaves the project, the system must prevent him from accessing any information about the project.

**- Smart home management:** In the context of smart home, the house owner has many connected objects inside his home. Each object is assigned a secret key relative to its role in the home system. If the home owner buys a new smart object and introduces it into his home, the new object should access with its new secret key any cipher-text that is eligible to, even if the latter is encrypted before the object's entry into the system.

Similarly, if ever the smart object is stolen or compromised, we would like to revoke all its access rights, so as, the thief or the attacker could not use object's information to access private information which are encrypted either before or after stealing the secret key.

### Security Requirements

In this section, we list the security requirements that our solution must meet, so as, it can be applied to the kind of applications that we described in Section 5.3.3.

1. **Definition 1: Attribute-Based Backward Accessibility.** After gaining new attributes, a user will be able to decrypt all old cipher-texts encrypted, even data encrypted before the change in the attributes set occurs, with a policy which is satisfiable by his new attributes set.

2. **Definition 2: Attribute-Based Forward Accessibility.** After gaining new attributes, a user will be able to decrypt all future cipher-texts that will be encrypted with a policy satisfiable by his new attributes set.

3. **Definition 3: Attribute-Based Forward Secrecy.** After losing one or many attributes, a user must have no access to *future* cipher-text decryptable with its previous private key if its new attributes set doesn't satisfy the encryption policy. In other words, once a user secret key is updated, the previous key is completely unusable.

4. **Definition 4: Attribute-Based Backward Secrecy.** After losing one or many attributes, a user must have no access to *old* cipher-texts decryptable with its previous private key if its new attributes set doesn't satisfy the encryption policy. In other words, once a user secret key is updated, the previous key is completely unusable.

5. **Collusion Resistance.** This is a very important property of Attribute Based Encryption. It means that the conspiracy of many non-authorized users for decrypting a cipher-text is useless, even if the union of their attributes sets satisfies the encryption policy of the cipher-text.

6. **Immediate Key Update.** All the changes made in a user's attributes set must take effect immediately. Revoking one or many attributes for a user, and/or adding new attributes to

his attributes set must done instantly following the decision to do that. The system must not wait for the update of users' keys. This property is very important in real time systems where changes about users' access rights occur unpredictably and frequently.

7. **User's privacy.** The attribute management mechanism must preserve user privacy. Information concerning a user like its attributes set must be kept secret from a third party even from the proxy. The data accessed by a user also must be kept hidden to other users.

### Basic idea

The proxy holds in its memory the list of all users' identities, the symmetric keys and secret key parts generated by the Attribute Authority. The table 5.3 shows how the proxy stores all these information in a data structure.

Table 5.3: Proxy data structure

| User identity | Symmetric key | Secret key part |
|---|---|---|
| 1 | $K_{p,1}$ | $D^{(1)}$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $i$ | $K_{p,i}$ | $D^{(i)}$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $N$ | $K_{p,N}$ | $D^{(N)}$ |

### Scheme

In this section we present our construction of CP-ABE to achieve proxy-based real-time attribute/key revocation under security requirements stated in section 5.3.3.

Figure 5.13 summarizes the different phases of our scheme and shows message exchanges during each phase.

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map (see Section 3.2.2)

The scheme consists of four primitives:

- **Setup**. The setup primitive is run by the Attribute Authority to generate a Public Key $PK$ and a Master Key $MK$. It chooses a bilinear group $\mathbb{G}_0$ of prime order $p$. Then, it chooses two random $\alpha, \beta \in \mathbb{Z}_p$. The primitive outputs $PK$ and $MK$:

$$PK = G_0, g, h = g^{\beta}, f = g^{1/\beta}, e\,(g,g)^{\alpha} \tag{5.17}$$

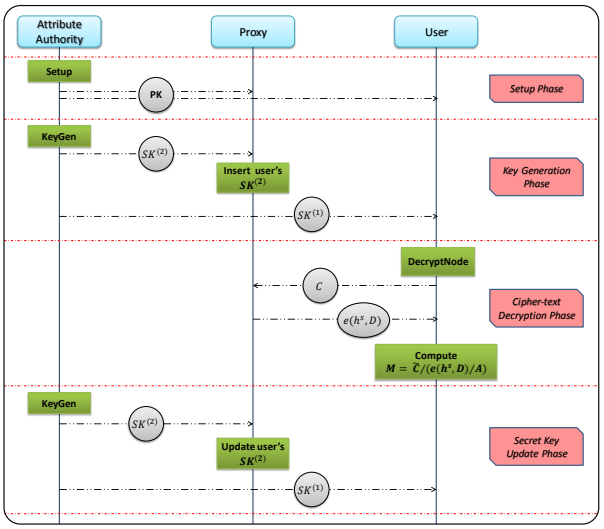



Figure 5.13: Sequence Diagram of different phases of our scheme

$$MK = (\beta, g^{\alpha}) \tag{5.18}$$

- **KeyGen**(MK,S). It is run by the Attribute Authority to generate a Secret Key for every user from its attributes set. It takes as input the user set of attributes $S$. The primitive chooses a random $r \in \mathbb{Z}_p$, and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$. It constructs the secret key exactly as in [11] (formula 5.19). Instead of giving it entirely to the user, the Attribute Authority splits it into two parts: the first one $SK^{(1)}$ contains all $D_j$ and $D'_j$ and it is sent to the corresponding entity/user. The second one $SK^{(2)}$ consists only of $D$ and it is sent to the proxy.

$$SK = (D = g^{(\alpha+r)/\beta}, \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j}) \tag{5.19}$$

$$SK^{(1)} = \left(\forall j \in S : D_j, D'_j\right). \quad SK^{(2)} = D. \tag{5.20}$$

If the Attribute Authority generates a private key for the first time, the proxy creates a new record in its users' table (Table 5.3) and fills it with user's information.

If the received private key is associated to an existing user in the table, the proxy updates only the field corresponding to private key part. Once the user key is updated, the part of the previous private key held by the corresponding user is useless. Indeed, during the decryption process, the user needs the secret key part $SK^{(2)}$ held by the proxy.

- **Encrypt**(PK,$\gamma$,M).

  The encryption primitive encrypts a message $M$ under the tree access $\gamma$. The algorithm first chooses a polynomial $q_x$ for each node $x$ (including the leaves) in the access tree $\gamma$. These polynomials are chosen in the following way in a top-down manner, starting from the root $R$. For each node $x$ in the tree, set the degree $d_x$ of the polynomial $q_x$ to be one less than the threshold value $k_x$ of that node, that is, $d_x = k_x - 1$.

  Starting with the root node $R$ the algorithm chooses a random $s \in \mathbb{Z}_p$ and sets $q_R(0) = s$. Then, it chooses $d_R$ other points of the polynomial $q_R$ randomly to define it completely. For any other node $x$, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses $d_x$ other points randomly to completely define $q_x$.

  Let $Y$ be the set of leaf nodes in $\gamma$. The ciphertext is then constructed by giving the tree access tree $\gamma$ and computing:

$$CT = \Bigg(\gamma, \tilde{C} = Me(g,g)^{\alpha s}, C = \{h^s\}_{PrPK}, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\Bigg) \quad (5.21)$$

  The plaintext of $h^s$ is encrypted with the Proxy Public Key, therefore, none else can get this value, even the user decrypting the message.

- **Decrypt**(CT,SK). The decryption primitive decrypts the ciphertext $CT$ using the private key $SK$ which is associated with a set $S$ of attributes. This primitive uses the *DecryptNode* function defined in [11], then sets $A = DecryptNode(CT, SK, r) = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$.

  In order to complete the decryption process the user needs the value of $e(h^s, D)$. So, the user sends the value of $C$ encrypted with the symmetric key $K_{p,u}$ to the proxy. The proxy is able to recover the value of $h^s$ by double decrypting the user's message, the first one by using symmetric key $K_{p,u}$, and the second one by using $PrSK$. Then, it computes $e(h^s, D)$ and sends it back to the user.

  Once the user receives $e(h^s, D)$, she/he can proceed to the decryption by computing the message $M$ this way:

$$\begin{aligned} \tilde{C}/\left(e(h^s, D)/A\right) &= \tilde{C}/\left(e\left(h^s, g^{(\alpha+r)/\beta}\right)/e(g,g)^{rs}\right) \\ &= M \end{aligned} \quad (5.22)$$

**Security Analysis**

In this section, we prove that our solution is secure and meets the security requirements defined in subsection 5.3.3.

**Proposition 1.** *In our scheme, the participation of the proxy in the decryption process is necessary. Users require the proxy to be able to decrypt any cipher-text.*

*proof:*

Our scheme, as we describe it, forces the user to solicit the proxy for computing $e\left(h^s, D\right)$ during the decryption. This is because the user is not aware of $D$. As the latter is a user secret key part which is sent to the proxy.

In other hand, we have to be sure that multiple solicitation of the proxy does not allow to a user to retrieve the value of $D$ or even guess the result $e(h^x, D)$ without the assistance of the proxy. (Where $h^x$ is a part of a cipher-text. a random number used to encrypt a message).

First, the FAPI-1 assumption presented in Section 5.3.2 prevents the user from computing the value of $D$ using the intermediate result $e(h^s, D)$ calculated by the proxy.

Furthermore, as the value of $h^s$ is encrypted with the proxy public key $PrPK$, it is hidden to users; only the proxy can get its value after decryption using its proxy secret key $PrSK$. The FAPI-2 assumption (Section 5.3.2) ensures also that the user cannot get $h^s$ from $e(h^s, D)$ which is computed by the proxy. Hence, it is not possible for any user to find a correlation between old $h^{s_i}$ $(i = 1, 2, ...)$ and a new $h^x$.

**Proposition 2.** *Our scheme achieves Attribute-Based Backward and Forward Accessibility and Attribute-Based Forward and Backward Secrecy as they are defined in Section 5.3.3.*

*proof:*

As soon as the Attribute Authority updates the user secret key and sends $SK^{(2)}$ to the proxy, and $SK^{(1)}$ to the concerned user, the latter is able to decrypt all old and future cipher-texts with an access policy satisfied by her/his new attributes set. Hence, Attribute-Based Backward and Forward Accessibility are verified (Definitions 1 and 2 Section 5.3.3).

On another side, the user's previous key is no longer usable as the part $SK^{(2)}$ is updated in the proxy side. Indeed, all what the user could do with it is to compute $e(g, g)^{rs}$ using her/his $SK^{(1)}$. The $r$ in the exponent is relative to the secret key, and it appears only in proxy's part $SK^{(2)}$. therefore, the result $e(g, g)^{rs}$ is unusable as it is randomized with $r$. On the other hand, Proposition 1 ensures that the user needs the proxy in the decryption process, and, as the proxy's secret part is updated, it is no longer possible for the user to use her/his previous attributes set to decrypt any cipher-text. Hence, Attribute-Based Backward and Forward Secrecy are verified (Definitions 3 and 4 Section 5.3.3).

**Proposition 3.** *Our scheme achieves collusion resistance property as defined in section 5.3.3.*

*proof:*

The CP-ABE scheme is constructed in a such way that does not allow collusion between two users having two different secret keys. Indeed a secret key is randomized with a random number $r$ (Equation 5.19). More information can be found in [11] about collusion resistance of CP-ABE.

As our scheme is based on CP-ABE, this property is also included.

**Proposition 4.** *Our solution ensures the user privacy (data and attributes set): The user's attributes set and data she/he wants to decrypt are hidden to the proxy and to any third party.*

*proof:*

All the communications between the user and the proxy are encrypted with symmetric shared key (Section 5.3.2: Assumption 1), therefore, any third party cannot have any information about the desired cipher-text to decrypted. This also remains true about the attributes set, as the Attribute Authority sends the secret elements ($SK^{(1)}$ and $SK^{(2)}$) securely to the user and the proxy respectively.

The part of the secret key sent to the proxy $SK^{(2)}$ contains only the element $D$. It reveals no information about the user's attributes set, therefore, the proxy cannot be aware of the user's privileges. Likewise, during the decryption process, the only information that the proxy could be aware of is $h^s$ which does not reveal much about the cipher-text $CT$ itself.

**Proposition 5.** *Our scheme does not allow to a curious proxy to use users' privileges to access data.*

*proof:*

The proxy possesses only a part $SK^{(1)} = D$ of the original CP-ABE user's secret key $SK$. This part all alone is useless during the decryption process as it needs the elements $D_j$ and $D'_j$ associated to the different attributes. Indeed, in CP-ABE, having only $SK^{(1)} = D$ is equivalent to an empty set of attributes $S$, and hence, no cipher-text could be decrypted with it.

### Performance Analysis

In this section, we discuss the performances of our scheme in terms of required storage capacity (for both the proxy and the user), decryption cost and secret key revocation cost. The encryption primitive is almost the same as the one of CP-ABE [11] and PIRATTE [43], with one extra symmetric encryption (For example AES algorithm).

We have chosen four (4) different pairings parameters ("*a*", "*a1*", "*d159*" and "*f*") to analyze the performances of our solution. Sizes of elements for different pairings parameters are given in the table 5.4.

| | a | a1 | d159 | f |
|---|---|---|---|---|
| $\mathbb{G}_1$ (bytes) | 132 | 264 | 44 | 44 |
| $\mathbb{G}_2$ (bytes) | 132 | 264 | 124 | 84 |
| $\mathbb{G}_t$ (bytes) | 132 | 264 | 124 | 244 |
| $\mathbb{Z}_p$ (bytes) | 24 | 132 | 24 | 24 |

Table 5.4: Size of elements for different pairings

**Size of the proxy table** Let $N$ be the number of the users managed in the system. The identifier of a user could be codified with $\log_2(N)$ bits. We assume that the symmetric key is a AES key codified in 128 bits. The size of $D$ ($D \in \mathbb{G}_2$) depends on the pairing parameters used in the implementation of CP-ABE (See Table 5.4).

In conclusion we have:

$$Size\,(Table) = (128 + |D| + \log_2(N)) \cdot Nbits. \tag{5.23}$$

Table 5.5: Required storage size for the proxy to store users' table

| Users number | Required storage size (kB) | | | |
|---|---|---|---|---|
| | a | a1 | d159 | f |
| 10 | 1.4494 | 2.7384 | 1.3712 | 0.9806 |
| 200 | 29.0929 | 54.8741 | 27.5304 | 19.7179 |
| 400 | 58.2346 | 109.7971 | 55.1096 | 39.4846 |
| 600 | 87.3947 | 164.7384 | 82.7072 | 59.2697 |
| 800 | 116.5668 | 219.6918 | 110.3168 | 79.0668 |
| 1000 | 145.7478 | 274.6540 | 137.9353 | 98.8728 |

The size of the table does not have to bother much as our protocol allows to spread the decryption assisting overhead (storage and computation) across multiple proxies. To do so, all the proxies will share the same proxy secret key $PrSK$ so they can decrypt the element $C$. Each user is assigned to only one proxy, and each proxy is assigned a set of users and it is responsible to assist them in their decryption processes. Thereby, the load of computation and storage upon one proxy is lightened.

Figure 5.14 shows the variation of the required storage size at the proxy with respect to the number of users in the system. For example, 960 users require 270 kB, 144 kB, 136 kB, and 98 kB in the cases of pairing parameters *a1*, *a*, *d156*, and *f* respectively.

**Size of User's Private Key** The size of the user's secret key in our scheme contains an integer $n$ (Encoded in four (4) bytes) representing the number of attributes, and $n = |S|$ couples of elements

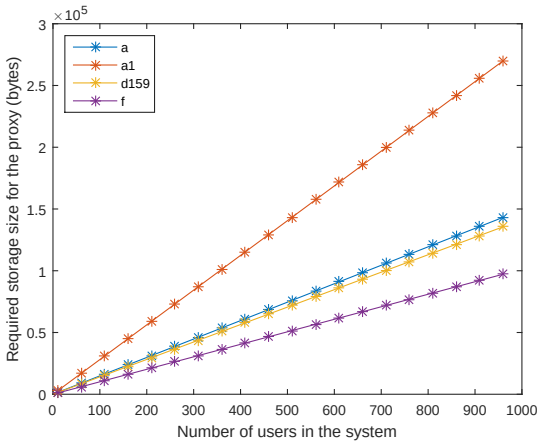



Figure 5.14: Required storage size with respect to number of users

$< D_j, D'_j >$ from $\mathbb{G}_2$ and $\mathbb{G}_1$ respectively. The average size of attributes is represented as $a$. CP-ABE $SK$ contains also an element $D$ from $\mathbb{G}_2$ (This element is stored by proxy in our solution). The private key in PIRATTE has in addition $n$ elements $D''_j$ from $\mathbb{G}_1$.

The table 5.6 summarizes the size of the user's secret key fro the three schemes depending on the pairing parameters.

Table 5.6: Size of the user's secret key

| | Secret key size (bytes) | | | |
|---|---|---|---|---|
| | a | a1 | d159 | f |
| CP-ABE [11] | $136 + (a + 264)n$ | $268 + (a + 528)n$ | $128 + (a + 168)n$ | $88 + (a + 128)n$ |
| Piratte [43] | $136 + (a + 392)n$ | $268 + (a + 792)n$ | $128 + (a + 212)n$ | $88 + (a + 172)n$ |
| Ours | $4 + (a + 264)n$ | $4 + (a + 528)n$ | $4 + (a + 168)n$ | $4 + (a + 128)n$ |

Figure 5.15 shows the comparison between original CP-ABE [11], PIRATTE [43] and our solution in terms of user's secret key size for different pairings parameters ("a", "a1", "d159", and "f"). The number of attribute $n$ is set to $50$, and the average size of attributes $a$ is set to $20$ bytes. We notice that, our solution has the lowest size for each pairing parameter.

**Decryption cost** The decryption process in our solution requires only one extra exchange (One sending and one reception) of messages between the decryptor and the proxy. As stated before, the exchanged data are encrypted before sending and decrypting after reception (symmetric encryption and decryption). As the overhead of symmetric cryptography is negligible compared to CP-ABE operations, we did not include it in the performance analysis. However, our solution allows the decryptor to save up one pairing operation as the latter is executed by the proxy.

The original CP-ABE [11] is not concerned by exchanged messages overhead as the decryption process is entirely run at the decryptor side (But attributes revocation is not implemented). As for PIRATTE [43], for every leaf node in the access tree, the user is required to send an element $C'_y$ of

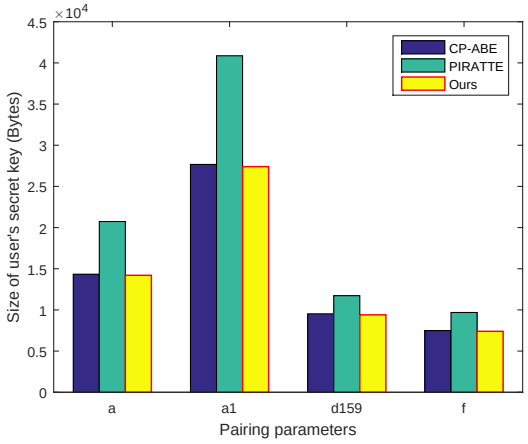



Figure 5.15: Size of user's secret key ($n = 50, a = 20$)

$\mathbb{G}_2$ and to receive from the proxy an element $C''_y$ of $\mathbb{G}_2$ and another of $\mathbb{Z}_p$. Table 5.7 summarizes the sizes of exchanged messages for the three solutions (Where $|Y|$ is the number leaf nodes in the access access of the cipher-text). It is obvious from Table 5.7 that our solution widely overcomes PIRATTE [43] in term of exchanged messages overhead. Indeed, in our solution, the sizes of messages stay constant regardless of the access tree. However, PIRATTE requires exchanging messages with sizes growing linearly with the number of leaf nodes in the access tree as it appears in Table 5.7.

Table 5.7: Size of exchanged messages during decryption

| | | Size of exchanged messages (bytes) | | | |
|---|---|---|---|---|---|
| | | a | a1 | d159 | f |
| CP-ABE [11] | Sent | – | – | – | – |
| | Received | – | – | – | – |
| PIRATTE [43] | Sent | $132 \cdot \|Y\|$ | $264 \cdot \|Y\|$ | $124 \cdot \|Y\|$ | $84 \cdot \|Y\|$ |
| | Received | $156 \cdot \|Y\|$ | $396 \cdot \|Y\|$ | $148 \cdot \|Y\|$ | $108 \cdot \|Y\|$ |
| Ours | Sent | 132 | 264 | 124 | 84 |
| | Received | 132 | 264 | 124 | 244 |

**Secret key update cost** The key revocation process costs exactly same as the key generation for the Attribute Authority, the proxy and the user. The running time at the Attribute Authority depends on the number of the attributes in the new attributes set.

The AA generates a new secret key associated to the new attributes set. Then, it sends the first part ($SK^{(1)}$) to the user, and the second one ($SK^{(2)}$) to the proxy. After receiving the key parts, both of the user and the proxy update the secret key parts. Once, the proxy does the update of the user's secret key part, the revocation takes effect.

One thing very important in our scheme is the fact that, attribute/key revocation does not affect any other non-concerned user, and it does not require re-encrypting data to guarantee the revocation.

Our scheme ensures attribute and/or key immediate revocation.

### 5.3.4 Solution 2: Backward and Forward Secrecy

#### Motivations and Application Cases

In this variant we tackle the attribute/user revocation issue of CP-ABE in some application cases which have specific properties and security requirements. In these applications, users gaining new attributes must be prevented from accessing old encrypted data. Likewise, a user losing some attributes will no longer be able to decrypt future cipher-texts requiring some of the lost attributes.

In other words, the validity period of a user's key is bounded by its delivery (beginning of the validity) and its revocation (end of the validity). A user's key can only decrypt cipher-texts which are encrypted during its validity periods.

We can cite "MultiCast Group Communication" as a kind of applications that require these security properties. Many applications are part of MultiCast Group Communications. For example: *Online Network Games*, *Video on Demand*, *Chat rooms*, *TVoD (Encrypted TV)*, etc.

**- Chat rooms:** Chat rooms applications require a confidentiality of communications. Users joining chat rooms gain attributes that allow them to decrypt exchanged messages. The chat room managing system must prevent new users from accessing old messages sent before their membership. Further, users quitting a chat room lose all access rights to communications between the members.

**- TVoD (Encrypted TV):** One of the applications that we target is Encrypted TV. Users subscribe to channels and programs. Users gain the access to the channels and programs only from the moment they subscribe (Backward secrecy). Likewise, when the validity of the subscription expires or the service provider decides to stop the subscription, the users lose the access to future programs (Forward secrecy).

#### Security Requirements

In this section, we present the security requirements that our mechanism must verify in order to be validated. The applications we target in this paper require specific security properties which are summarized below.

- **Backward Secrecy.** A user receiving new secret key with new attributes is not able to use it to decrypt old messages.
- **Forward Secrecy.** A user receiving a new secret key after losing some attributes has no access to futures cipher-text requiring the lost attributes even if these cipher-texts' policies are satisfied by a previous secret key.

- **Collusion resistant.** This is a very important property of Attribute Based Encryption. It means that the conspiracy of many non-authorized users for decrypting a cipher-text is useless, even if the union of their attributes sets satisfies the encryption policy defined for the cipher-text.
- **Immediate Key Update.** When a user's attributes set is updated (adding and/or removing attributes), a new secret key must be constructed based on the new user's attributes set and must be delivered for that user immediately.
- **User Privacy.** The attribute management mechanism must preserve user privacy. Information concerning a user like its attributes list must be kept secret from a third party. The data accessed by a user also must be hidden from other users.

### Basic idea

We assume the existence of a proxy, having less resource constraints, that will assist entities during the decryption process. The latter must keeps information about users. The data structure in figure 5.8 shows which information the proxy keeps about a given user $U_i$.

| Secret key part | Delivery date |
|---|---|
| $D^{(1)}$ | $DD_1$ |
| $\vdots$ | $\vdots$ |
| $D^{(i)}$ | $DD_i$ |
| $D^{(i+1)}$ | $DD_{i+1}$ |
| $\vdots$ | $\vdots$ |
| $D^{(n)}$ | $DD_n$ |

Table 5.8: Proxy data structure

In the first column we have the secret key part $D_i$ and the corresponding delivery date $DD_i$ is in the second column.

We split the secret key into two parts, the first one, $SK^{(1)}$ which contains all elements related to the attributes is sent to the user. The second part $SK^{(2)}$ which represents the element $D$, is sent to the proxy along with the time of key update (Delivery date) $DD$. The proxy maintains all the history of users' keys.

We bind the $h^s$ with the time of encryption $T_{enc}$ in the ciphertext $CT$ by encrypting them with the proxy public key $PrPK$. We finally get $C = \{h^s, T_{enc}\}_{_PrPK}$.

During the decryption, the user has to solicit the proxy to completely recover the plain-text. She/he sends $C$ to the proxy. The latter decrypt is using $PrSK$ and then, extract the $T_{enc}$. Using $T_{enc}$ the proxy finds the element $D_i$ to use. Then, it computes $e(h^s, D_i)$ and sends it back to the user. To find the $D_i$, the proxy can do it by dichotomy. The $i$ of $D_i$ must satisfy this property:

$$DD_i \leq T_{enc} < DD_{i+1} \tag{5.24}$$

**Scheme**

In this section, we describe in detail the different primitives and how our solution achieves attribute revocation.

**Setup.** The setup primitive generates the public key $PK$ and the master key $MK$ of the system. It is run by the Attribute Authority during the bootstrap phase. The primitive chooses a bilinear group $\mathbb{G}_0$ of prime order $p$ with generator $g$. Then it chooses two random exponents $\alpha, \beta \in \mathbb{Z}_p$. The public key is published as:

$$PK = G_0, g, h = g^{\beta}, f = g^{1/\beta}, e\left(g, g\right)^{\alpha}. \tag{5.25}$$

and the master key is:

$$MK = (\beta, g^{\alpha}). \tag{5.26}$$

**KeyGen**(MK,S).

This primitive is run by the Attribute Authority to generate users' secret keys. It takes as input the master key $MK$ and user's attributes set $S$. It outputs two parts $SK^{(1)}$ and $SK^{(2)}$. The first one is given to the corresponding user, and the other one is sent to the proxy.

The algorithm first chooses a random $r \in \mathbb{Z}_p$, and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$. It computes two parts as:

$$SK^{(1)} = \left(\forall j \in S : D_j = g^r \cdot H\left(j\right)^{r_j}, D'_j = g^{r_j}\right) \tag{5.27}$$

and

$$SK^{(2)} = D = g^{(\alpha+r)/\beta} \tag{5.28}$$

**Encrypt**(PK,$\gamma$,M).

The encryption primitive encrypts a message $M$ under the tree access $\gamma$. The algorithm first chooses a polynomial $q_x$ for each node $x$ (including the leaves) in the access tree $\gamma$. These polynomials are chosen in the following way in a top-down manner, starting from the root $R$. For each node $x$ in the tree, set the degree $d_x$ of the polynomial $q_x$ to be one less than the threshold value $k_x$ of that node, that is, $d_x = k_x - 1$.

Starting with the root node $R$ the algorithm chooses a random $s \in \mathbb{Z}_p$ and sets $q_R(0) = s$. Then, it chooses $d_R$ other points of the polynomial $q_R$ randomly to define it completely. For any other node $x$, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses $d_x$ other points randomly to completely define $q_x$.

Let $Y$ be the set of leaf nodes in $\gamma$. The cipher-text is then constructed by giving the tree access tree $\gamma$ and computing:

$$CT = \Big(\gamma, \tilde{C} = Me(g,g)^{\alpha s}, C = \{h^s, T_{enc}\}_PrPK, \\ \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\Big) \quad (5.29)$$

**Decrypt**(CT,SK).

During the decryption process, an decryptor uses the *DecryptNode* function defined in [11] to compute $A = DecryptNode(CT, SK, r) = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$. Where $r$ represents the root node of the access tree $\gamma$ defined for $CT$.

The decryptor requires $e(h^s, D)$ value to proceed the decryption, he sends $C$ to the assisting proxy. The latter uses proxy private key $PrPK$ to decrypt the message content. Then it choose the adequate $D$ to use according to $T_{enc}$. The proxy compute $R = e(h^s, D)$ and send it back encrypted to the decryptor.

Now, the decryptor can retrieve the original message this way:

$$\begin{aligned} \tilde{C}/(e(R,D)/A) &= \tilde{C}/(e(h^s,D)/A) \\ &= \tilde{C}/\left(e\left(h^s, g^{(\alpha+r)/\beta}\right)/e(g,g)^{rs}\right) \\ &= M \end{aligned} \quad (5.30)$$

The decryption primitive succeeds if and only if the set of attributes associated with $SK$ satisfies the access policy $\gamma$ defined for $CT$. Otherwise, the primitive fails and returns nothing.

### Security Analysis

In this section, we give proofs that our revocation mechanism meets the security requirements defined in Section 5.3.4

**Proposition 1.** *Our scheme guarantees Backward secrecy.*

*Proof.*

Once the Attributes Authority updates the user's secret key, the new secret key is usable only for cipher-texts with a timestamps greater than it delivery date.

We may believe that the user could cheat the proxy by constructing a fake $C$ element. He could choose a timestamps in the interval of an old key, but the value of $h^s$ is hidden to user. Hence, he cannot forge a fake $C$ element of the cipher-text.

**Proposition 2.** *Our scheme guarantees Forward secrecy.*

*Proof.*

Once the user's secret key is updated: the user receives elements related to the new attributes set, and the proxy receives from the Attribute Authority the $D_{i+1}$ element at $DD_{i+1}$. The previous user's key cannot be used to decrypt messages encrypted after $DD_{i+1}$.

Th user cannot forge a fake $C$ element with an old time of encryption, as the latter is composed of $h^s$ and the time of encryption $T_{enc}$, all encrypted with the proxy public key $PrPK$. Indeed, the user is not aware of the $h^s$ as it is hidden by encryption.

**Proposition 3.** *Our scheme prevents users from collusion.*

*Proof.*

The collusion resistance is a property ensured the ABE scheme construction. Thanks to the random elements $r$, $r_j$, secret keys cannot be used together as they are randomized. For more information about collusion ressistance property, we invite the reader to take a look at the original paper [11].

As our solution is based on CP-ABE scheme, this property is also verified.

**Proposition 4.** *Our scheme ensures immediate users' keys update.*

*Proof.*

Once the Attribute Authority decides to update a user's secret key, it just had to generate a new CP-ABE secret key and as detailed in Section 5.3.4 and securely send the two parts to the concerned user and the proxy. Since then, the user's secret key is effectively updated and his the previous secret key is no longer usable for future cipher-texts.

**Proposition 5.** *Our scheme ensures users' privacy.*

*Proof.*

The user's privacy in question here is about his attributes set and the cipher-texts he intend to decrypt. Indeed, the proxy has no idea about the list of attributes of any user, as he possesses only the element $SK^{(2)} = D$ related to the secret key which gives no information about the attributes set.

Likewise, during the decryption process, the proxy receives only the two elements $\{h^s, T_{enc}\}$ encrypted with his public key $PrPK$. These two elements don't say much about the cipher-text itself except the time of encryption.

**Performance Analysis**

In this section, we analyze the performances of our solution in terms of computation and storage costs. Then, we give a highlight on how we can speed up the search at the side of the proxy.

**Computation performance** **Setup and Key Generation primitives:** Our solution executes the *setup* and *keygen* primitives exactly as the original CP-ABE [11] except that it splits the secret key into two parts and sends them to the user and the proxy. It does not require more computation overhead.

**Encrypt primitive:** The encryption primitive of our solution is pretty similar to the original one, it just adds a timestamps $T_{enc}$ to the cipher-text and replaces the element $C$ by the concatenation of $h^s$ and $T_{enc}$ all encrypted with the proxy public key $PrPK$. Our solution does not generate much overhead for the data owner.

**Decrypt primitive:** The Decrypt primitive of our solution requires an exchange between the user and the proxy. The user sends the element $C$ whose size equals $|\mathbb{G}_1| + |T_{enc}|$ and receives and element from $\mathbb{G}_T$.

The proxy has to look for the adequate $D_i$ in the proxy table (Formula 5.24), it can use the dichotomy method to speed up the search. The user also has to find the adequate $SK^{(1)}$ to use depending on $T_{enc}$. He also can use the dichotomy method.

**Storage performance** The storage overhead is an inherent drawback of our solution, as the proxy and the user have to store all old secret key parts. For the user, he can keep only the most recent key part $SK^{(1)}$ if he is sure that he does not need to access old messages.

## 5.4 Conclusion

Attribute/Key management is a keystone issue in CP-APE because of low efficiency of existing attribute revocation techniques. Indeed, existing solutions induce great side effect after each attribute revocation. The side effect induces rekeying and/or re-assignment of attributes to all users.

In this chapter we have proposed solutions for key/attribute revocation for CP-ABE in the context of IoT where efficiency is of a paramount importance.

First we proposed a batch-based version for CP-ABE to achieve attributes revocation. We proposed to split time axis into intervals (time slots) and to send only the necessary key parts to allow refreshing the secrets keys. An analysis is conducted on the way to choose the best time slot duration in order to maximize system performances and minimize average waiting time.

Then, we considered practical application scenarios in which the Attribute Authority knows beforehand all start dates and durations of all attributes validity periods of all entities in the system,

and proposed a scheme supporting attribute revocation. The solution we proposed induces zero delay and a minimum of generated secret key parts.

Finally, we considered application scenarios requiring immediate revocation of attributes and associated privileges without beforehand knowledge of attributes revocation schedule. Under those assumptions, we have proposed a proxy-based ABE mechanism with attribute/key revocation. Users' secret keys are split into two parts, the first one is sent to the user himself, the second one is stored by the proxy. The latter assist users during decryption process. The key update consists of generating a new secret key for the concerned user. Once the proxy receives its corresponding part of the new secret key, the revocation takes effect immediately. Our solution achieves efficiently immediate attributes/key revocation without affecting not concerned users.

# Chapter 6

# Threshold grouping proof in IoT

## 6.1 Introduction

Yoking-proof is a security concept which was introduced by Juels in 2004 [46]. It aims to provide a proof that a pair of RFID tags has been scanned simultaneously. Although it originally addresses RFID systems, yoking-proof can be considered for IoT applications. For example, the manager of an emergency center might want to ensure that the ambulance doesn't leave the garage without a mobile scanner. In its basic description, yoking-proof addresses only a pair of devices. Given the necessity of considering this concept for a group of more than two nodes, the notion of grouping-proof was proposed as a generalization of the yoking-proof concept [59][15]. However, existing yoking/grouping schemes are mostly designed for RFID systems. Characteristics of IoT applications introduce other requirements that have to be considered by yoking/grouping schemes.

In this chapter, we propose a new yoking/grouping scheme for IoT applications, based on the powerful security protocol (CP-ABE). We propose the notion of threshold in grouping schemes that consists in proving the simultaneous presence of at least $k$ entities. Moreover, we introduce the concept of entity importance to give to the presence of a particular object more importance in the grouping proof.

## 6.2 Yoking/Grouping Proof Techniques

The concept of yoking-proofs was proposed by Juels as means to prove that a pair of RFID tags has been present simultaneously [46]. The proposed technique uses the reading device to generate a proof that can be off-line verifiable by a third-party (verifier).

Several proposals and critics have been made on the yoking-proof protocol. Bolotnyy and Robine raised a security concern in [15]. Indeed, Jules's protocol reveals tags identifier which creates a privacy problem when the reader is untrusted. The authors introduced a new problem formulation called anonymous-yoking which requires preserving privacy in the yoking-proof protocol.

In [59] authors pointed out that Jules's protocol is vulnerable to replay attack and a yoking-proof can be obtained with one RFID tag. The authors proposed a version that introduces a timestamp generated by a database to prevent from replay attack. The authors proposed also a generalization

of the yoking-proof protocol to a group of tags. In their scheme, the reader sends the timestamp to all tags participating to the protocol, which is used to parameterize the proof. However, because timestamp increases sequentially, an untrusted reader can take a future timestamp value, use it on one tag and wait that the database sends this value to exploit the old result generated by the tag with the current results of the rest of the group, which breaks the simultaneity character of the yoking proof. Although Piramuthu proposed a solution based on the use of a random number instead of a timestamp [55], an attacker can attempt a brute force attack method, especially if the range of the random number is not large (i.e. the attacker uses all possible numbers and stores results). This issue was addressed by Cho et al. in [21], where the authors proposed a solution based on dividing the random number into two numbers to raise complexity of the brute force attack in terms of storage space. But this will work well as long as attackers have limited resources. Another technique can be found in [26], where authors use so random number as RFID tags. However, the proposed solution reveals tags identifiers and creates a privacy problem.

In the same context, Bolotnyy and Robine proposed in [15] a solution based on the construction of a circular chain while polling tags. The purpose behind the use of a chain is to ensure that an untrusted reader will not be able to mount a replay attack or generate a proof if it breaks the chain. However, when dealing with a large set of tags, the completion of the proof would take a time and an adversary can take a tag once it is interrogated, breaking therefore the simultaneity character. This issue was tackled by Fuentes et al. in [24]. The authors proposed an approach based on dividing the set of devices (the authors consider IoT devices) into several subsets with low cardinality and poll each subset in unpredictable manner. The scheme operates in a number of rounds in such a way that different subsets are rebuilt in each round, which reduces the chance that an adversary takes a device without corrupting the proof. However, this kind of solution would increase the execution time of the proof.

Furthermore, in all the aforementioned propositions, the size of the generated proof increases proportionally with the size of entities. This could create a storage problem for the verifier as it receives the proof. The same can be said about the stored keys (each entity shares its secret key with the verifier so the latter can verify the proof).

In this chapter, we propose a new yoking/grouping proof system based on Ciphertext Policy Attribute-Based Encryption (CP-ABE). We use CP-ABE to encrypt a message in such a way that it can be decrypted only if at least $k$ nodes are simultaneously present. For the best of our knowledge, this is the first work that tackle the grouping proof from this perspective.

## 6.3 CP-ABE based Threshold Grouping proof

### 6.3.1 Overview

The main idea of our solution is to use Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [11] mechanism to implement our threshold grouping proof scheme in such a way to encrypt a secret so

the latter can be decrypted only if at least $k$ entities are simultaneously present. Fist, the Attribute Authority generates a secret key $SK$ associated with the list of attributes $S$ (formulas 6.1). Where $N$ is the number of entities in the group.

$$S = \{attribute_1, attribute_2, \cdots, attribute_N\} \tag{6.1}$$

The couple of elements $(D_j, D'_j)$ (See Section 3.2.3) associated to the attribute $j$ is sent to the entity $j$, where $1 \leq j \leq N$. The element $D$ of the secret key is sent to a proxy who is intended to recover the secret.

A secret message is encrypted with the access policy (Formula 6.2)

$$\gamma = k\text{-out-of-}N\ (attribute_1, attribute_2, \cdots, attribute_N) \tag{6.2}$$

Where $k$ is the threshold ($1 \leq k \leq N$).

Our solution allows the use of many secrets, each one of them with a different threshold. This enables making different proofs with different thresholds.

### 6.3.2 Network Model

- **Entities:** we consider a group of $N$ entities with a predefined threshold $k$.
- **Proxy:** it is responsible of relating the group of entities in order to recover a secret.
- **Attributes Authority:** it is a special entity in the system. The Attribute Authority is responsible for configuring the system by creating Public and Master keys which are used after that for encrypting messages and creating secret keys.
- **Verifier:** which is responsible for generating the secret and verifying the proof.

### 6.3.3 Security requirements

- Collusion of any $k$ or more entities of the group makes the secret message $M$ easily computable.
- Collusion of any $k-1$ or fewer entities of the group leaves the secret message $M$ completely undetermined.
- Group entities must disclose no information about their secret elements during the secret recovery process.

- The presence of entities must be in the same time (within a short interval).
- An adversary or an untrusted reader must be prevented from knowing if an object with a given identity is present during the proof.

### 6.3.4 Construction

For the sake of simplicity we will consider a single group of entities. It is easy to generalize our scheme to many groups (A simple way to do that is explained in Section 6.3.6). Let suppose we want to construct the proof for a group of $N$ entities. We can split our protocol into three main phases: *System configuration* phase, *Sharing secret* phase and *Recovering secret* phase.

**System configuration**

The Attribute Authority starts by running the *Setup* primitive of CP-ABE to generate a public key $PK$ and a master key $MK$. The Attribute Authority chooses a bilinear group $\mathbb{G}_0$ of prime order $p$ with generator $g$. Next it will choose two random exponents $\alpha, \beta \in \mathbb{Z}_p$. The public key is published as:

$$PK = (\mathbb{G}_0, g, h = g^{\beta}, e(g,g)^{\alpha}) \tag{6.3}$$

and the master key $MK$ is $(\beta, g^{\alpha})$.

This operation is executed once at the beginning of the system configuration.

Given a group of $N$ entities $GE = \{E_1, E_2, \cdots, E_N\}$, the Attribute Authority executes the *KeyGen* primitive to generate a secret key $SK$ associated with the set of attributes $S$ (Formula 6.1). We suppose here that each attribute $attribute_i \in S$ is hold by the entity $E_i$.

It first chooses a random $r \in \mathbb{Z}_p$, and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$. Then it computes the key as

$$SK = (D = g^{(\alpha+r)/\beta}, \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j}) \tag{6.4}$$

The element $D$ of the key $SK$ is given to the proxy as it is responsible for doing the secret recovery process. Each couple $(D_j, D'_j)$ is given to the corresponding entity $E_j$ of the group holding the $attribute_j$. Figure 6.1 illustrates the system configuration phase.

**Sharing secret**

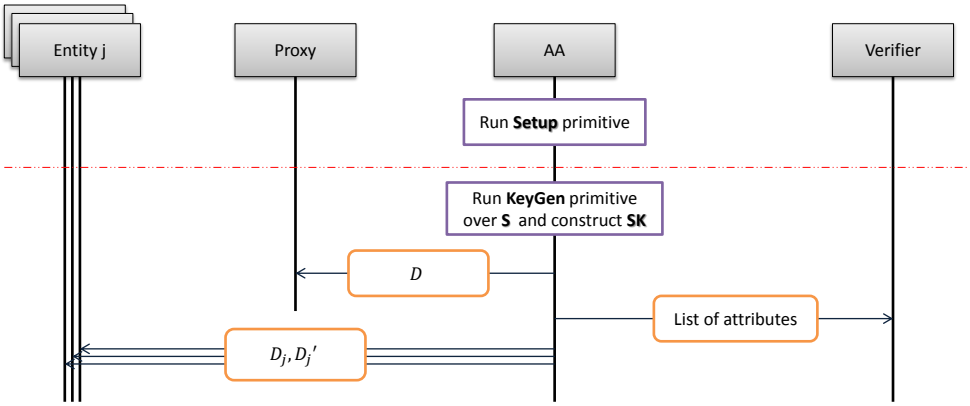



Figure 6.1: System configuration phase

When the grouping proof is needed, the verifier starts this phase and encrypts a secret $M$ with a particular access policy $\gamma$ ($k$-out-of-$N$ ($attribute_1, attribute_2, \cdots, attribute_N$)). The access tree associated with $\gamma$ has a node root with a threshold value $k_R = k$. The root node $R$ has $N$ children, all of them are leaves. Each leaf node is associated with one attribute $attribute_i$. $S$ is the set of leaf nodes.

The Encryption process begins with choosing a polynomial $q_x$ for each node $x$ in the tree $\gamma$. These polynomials are chosen in the following way in a top-down manner, starting from the root node $R$.

The degree of the polynomial corresponding to the root node is $d_R = k_R - 1 = k - 1$. All other polynomials associated to leaf nodes are with a degree equal to $d_x = k_x - 1 = 0$.

Starting with the root node $R$ the algorithm chooses a random $s \in \mathbb{Z}_p$ and sets $q_R(0) = s$. Then, it chooses $d_R$ other points of the polynomial $q_R$ randomly to define it completely.

For each leaf node $x$, the algorithm sets $q_x(0) = q_{parent(x)}(index(x)) = q_R(index(x))$.

The secret message is constructed as follows:

$$CT = (\gamma, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \forall y \in S : C_y = g^{q_y(0)}, C'_y = H\left(att\left(y\right)\right)^{q_y(0)}) \quad (6.5)$$

Once it is done, the verifier sends the secret message to the proxy and arms a timer. If the proxy could not recover the secret before the release of the timer, the verifier consider that the proof can not be constructed and the process stops.

**Recovering secret**

To recover a secret message $M$ shared with the group of entities, the proxy starts by requesting assistance from the group entities. If $k$ or more entities respond, the proxy can proceed with the recovery process, otherwise the secret recovery stops (Figure 6.2).

The proxy sends to each entity $E_j$ in the group the couple $(C_j, C'_j)$ of the ciphertext $CT$. Entities wishing to participate in the secret recovery process use their received values from the Attribute Authority to compute:

$$\begin{aligned} F_j &= \frac{e(D_j, C_j)}{e(D'_j, C'_j)} \\ &= \frac{e(g^r \cdot H(i)^{r_j}, h^{q_j(0)})}{e(g^{r_j}, H(j)^{q_j(0)})} \\ &= e(g,g)^{rq_j(0)} \end{aligned} \tag{6.6}$$

We suppose that the proxy receives at least $k$ intermediate results ($e(g,g)^{rq_j(0)}$ $(1 \leq j \leq N)$). Let $Q$ be the set of attributes corresponding to participating entities. $P$ is a subset of $Q$ that has a cardinal number equals to $k$. We have $P \subseteq Q \subseteq S$ and $|Q| \geq |P| = k$. The proxy proceeds with the decryption process by computing:

$$\begin{aligned} A &= \prod_{z \in P} (e(g,g)^{r \cdot q_z(0)})^{\Delta_{i,P'}(0)} \\ &= \prod_{z \in P} (e(g,g)^{r \cdot q_{parent(z)}(index(z))})^{\Delta_{i,P'}(0)} \\ &= \prod_{z \in P} (e(g,g)^{r \cdot q_R(i) \cdot \Delta_{i,P'}(0)}) \\ &= e(g,g)^{r \cdot \sum_{z \in P} q_R(i) \cdot \Delta_{i,P'}(0)} \\ &= e(g,g)^{r \cdot q_R(0)} \\ &= e(g,g)^{rs} \end{aligned} \tag{6.7}$$

Where $P' = \{index(z) : z \in P\}$, $i = index(z)$, and $\Delta_{i,P'}(0)$ is the Lagrange Coefficient [1].

The algorithm now recovers the secret by computing

$$\begin{aligned} \tilde{C} / \left( e\left(C, D\right) / A \right) &= \tilde{C} / \left( e\left(h^s, g^{(\alpha + r)/\beta}\right) / e\left(g, g\right)^{rs} \right) \\ &= M \end{aligned} \tag{6.8}$$

Once recovered, the proxy sends back the secret to the verifier which issued the proof.

[1] $\Delta_{i,S}\left(x\right) = \prod_{j \in S, j \neq i} \left(x - j\right) / \left(i - j\right).$

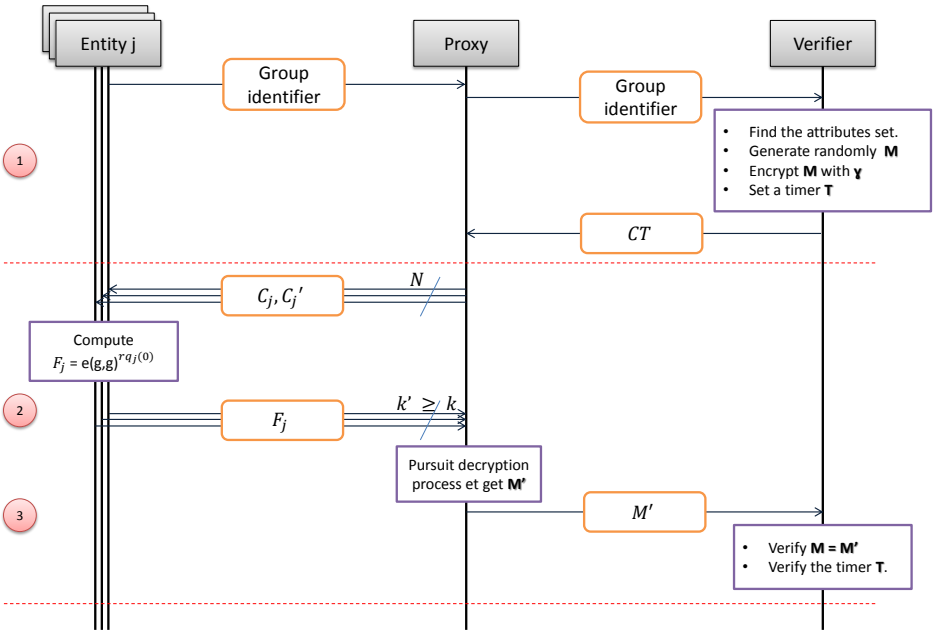



Figure 6.2: Recovering secret process

### 6.3.5 Group dynamics

Our solution allows easily and efficiently adding and removing entities to and from the group.

**Adding entities to the group.** In order to add an entity $E_{N+1}$ to the group of entities, the Attribute Authority must generate parts of the secret key $D_{N+1}$, $D'_{N+1}$ related to the attribute $attribute_(N+1)$ and send them to the new entity. To be able to do that, the Attribute Authority must have saved the random element $r$ related to the CP-ABE secret key's group, thus, it has just to pick a random number $r_{N+1}$ from $\mathbb{Z}_p$ and compute the couple $(D_{N+1}, D'_{N+1})$ this way:

$$\begin{cases} D_{N+1} = g^r \cdot H\left(attribute_(N+1)\right)^{r_{N+1}} \\ D'_{N+1} = g^{r_{N+1}} \end{cases} \tag{6.9}$$

Once an entity is added to the group, the Verifier will add the corresponding attribute to the access tree, so as, the new entity could participate to the decryption process.

The access tree will henceforth be like:

$$\gamma = k\text{-out-of-}(N+1)\Big(attribute_1, attribute_2, \cdots, attribute_N, attribute_(N+1)\Big) \tag{6.10}$$

**Removing entities from the group.** In order to remove an entity $E_e$ from the group of entities it is just enough to inform the Verifier that the entity is no longer part of the group, and he will remove its corresponding attribute $attribute_e$ from the access tree. The new access tree will be:

$$\gamma = k\text{-out-of-}(N-1)\Big(attribute_1, \cdots, attribute_(e-1), attribute_(e+1), \cdots, attribute_N\Big) \quad (6.11)$$

The proxy should be aware too as it is responsible of interrogating the entities during the decryption process.

As the attribute corresponding to the removed entity does not appear in the access tree, the entity could not participate to the decryption process even if it still has a part of the secret key.

### 6.3.6 Managing many groups of entities

The proposed scheme works well for one group of entities. Managing several groups of entities with the same system configuration requires to define a strategy of naming the attributes used in generating shared secret keys. A simple way to do that, is to associate an identifier $j$ for each group of entities. The set of attributes used for the $j^{th}$ group of entities will be $\{attribute_ < j > _1, attribute_ < j > _2, \cdots, attribute_ < j > _N_j\}$. Where $N_j$ is the number of entities in the $j^{th}$ group.

### 6.3.7 Security Analysis

Once a secret message $M$ is encrypted with the policy $\gamma$ (See formula 6.2) it is not possible to any third party to recover its value (None has a secret key with an attribute set satisfying $\gamma$).

As the threshold defined for this cipher-text is $k$ out of $N$ attributes, the collusion of any $k-1$ or less of entities belonging to the targeted group is pointless. Indeed, CP-ABE scheme prevents decrypting a cipher-text when the access policy is not satisfied by the attributes set of the secret key used. This property of our scheme meets the second security requirement (Section 6.3.3). However, The collusion of at least $k$ entities will end up by decrypting the cipher-text as it is detailed in Section 6.3.4.

An entity participating in the secret recovery does not reveal any information about the secret elements that she received from the Attributes Authority. Indeed, she computes $e(g,g)^{rq_j(0)}$, and sends it back to the proxy for recovering the secret. It is obvious that based on this information, neither the proxy nor a third party can get any information about $D_j = g^r \cdot H(j)^{r_j}$ or $D'_j = g^{r_j}$.

During the grouping process, objects don't use their identifiers to generate the proof. Even knowing object's attribute the proxy cannot know its identifier since attributes have no direct link with objects to whom they are associated. Thus, an adversary or an untrusted reader is unable to know if a given object is present during the proof.

The verifier uses a timer to ensure the presence of entities in the same time. If the secret could not be decrypted within a short time, the proof will not be constructed.

To attempt a replay attack with an untrusted proxy, the latter has to return the right secret message to the verifier. However, since the secret is encrypted with CP-ABE, the proxy cannot have access. Even brute force attack is difficult to consider because the proxy does not know what to return to the verifier.

### 6.3.8 Advantages

- **Variability of the threshold $k$:** The value of the threshold is determined at the time of encryption. Which means that, for each secret (cipher-text) we can define a different threshold with the same configuration of the system. For the same group, it is possible to perform multiple proofs with different thresholds without reconfiguring the system.

- **Variability of users' importance:** Our scheme has the ability to give to one entity or many of entities more importance than others. In other words, the participation of an entity $E_i$ in the secret recovery process is equivalent to the participation of many (two or more) other entities with less importance. This could be achieved by associating to important entities a number of attributes proportional to their importance in the group of entities.

- **No secret information disclosure:** Users involved in secret recovering do not disclose any information about their secret elements. They only do computations using their secret elements and send intermediate results that are necessary in the secret recovery process.

- **Possibility of adding and removing easily entities to/from the group:** Once the system configuration is done, it is still possible to add/remove easily entities to/from the group. The processes of adding and removing entities is explained in Section 6.3.5.

- **One Setup Many Use:** Once the system is setup, many proofs can be obtained from the configuration. Indeed, we can decrypt as much as we want with the secret key shared between different objects of the group without redoing the system configuration again.

We present in Table 6.1 a comparison between our proposal approach with previously presented related works. The solution we propose is resistant against replay and brute force attacks. It preserves privacy and ensures the Simultaneity presence of tags.

Table 6.1: Comparison between approaches

| | *Privacy* | *Replay attack* | *Brute force attack* | *Simultaneity character* | *Time with scalability* |
|---|---|---|---|---|---|
| [46] | - | - | - | - | / |
| [59] | - | + | - | - | + |
| [55] | + | + | - | - | + |
| [21] | + | + | - | - | + |
| [15] | + | + | + | - | - |
| [24] | + | + | + | + | - |
| *Ours* | + | + | + | + | + |

### 6.3.9 Performance Analysis

Performing our proposal scheme requires encrypting a random message (by the verifier) and recovering the secret (by the entities and the proxy). Number of operations that costs each node during each phase is summarized in Table 6.2.

Table 6.2: Performance Analysis: Number of operations

| | | **Nb. Pairing** | **Nb. Exp.** | **Nb. Mul.** | **Nb. Div.** |
|---|---|---|---|---|---|
| *System Configuration Phase* | AA | 1 | $(3) + (3N + 1)$ | 1 | 1 |
| *Sharing Secret Phase* | Encryptor | - | $2(N + 1)$ | 1 | - |
| *Secret Recovery Phase* | Each Entity | 2 | - | - | 1 |
| | Proxy | 1 | $k$ | $k - 1$ | 2 |
| *CP-ABE decryption* | Decryptor | $2N + 1$ | $k$ | $k - 1$ | $N + 2$ |

#### Encryption

CP-ABE is known to be very complex and expensive in terms of computation and energy consumption, especially the encryption primitive. Indeed, there are $2 * N + 2$ exponentiation and one (1) multiplication to do during the encryption process. However, in our proposal scheme this operation is performed by the verifier which commonly has considerable resources. When this primitive is executed by a resource-constrained device, this could drain quickly it battery and therefore reduce considerably its lifetime. However, In [63] a solution has been proposed to reduce this computation overhead and the overall energy consumption. It leverages the heterogeneity of the IoT environment to delegate costly operations namely exponetiations to more powerful trusted devices in the neighborhood.

Other solutions were proposed in [12] [41]. These solutions consist on splitting the encryption primitive into two steps. The first one which does the most of heavy operations is executed when the device is online (using harvested energy or line power). The second step is done offline when

the device is not connected to an energy source (Offline), it continue executing the rest of the encryption primitive.

**Secret Recovery**

Our scheme allows to split the overhead of CP-ABE decryption primitive between the group of entities and the proxy. Therefore, each entity computes only two pairings and one division, rather than $2N + 1$ pairing and $N + 2$ divisions. The exponentiation and the multiplication operations are performed by the proxy which has commonly significant resources. This particularity of our schema is very interesting when it is implemented in resource-constrained devices.

We have made a comparison between our approach and the original CP-ABE in term of decryption primitive performances. we considered an application case where the number of entities in the group $N = 20$, the threshold for decryption $k = 10$. The numbers of operations executed by an entity in both original CP-ABE and our approach are illustrated in Figure 6.3.

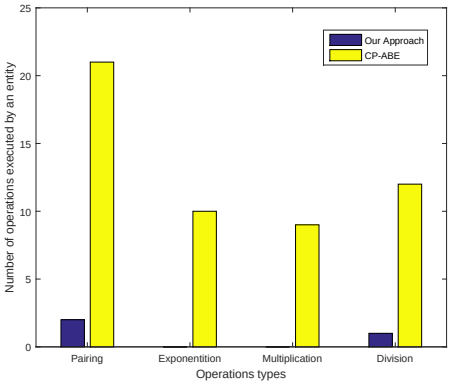



Figure 6.3: Number of operations executed during the decryption process by each entity

We notice that our approach reduces considerably the number of operations to execute during the decryption process. Indeed, it squarely remove exponentiations and multiplications as they are executed alongside the proxy. It also reduces the number of pairing operations and divisions to two (2) and one (1) respectively.

**Secret Key Storage**

Sizes of all elements used PBC library [4] are given in Table 6.3. We recall that these sizes are specific to the pairing parameters in the file *"f.param"*.

Table 6.3: Elements Sizes

| Group | Size (bytes) |
|---|---|
| $\mathbb{G}_1$ | 44 |
| $\mathbb{G}_2$ | 84 |
| $\mathbb{G}_T$ | 244 |
| $\mathbb{Z}_p$ | 24 |

Table 6.4 shows the size of secret elements sizes of both entity and proxy for our scheme, and for original CP-ABE. $N_g$ notices the number of considered group and $N_i$ denotes the number of entities in the $i^{th}$ group ($1 \leq i \leq N_g$). We notice that an entity in our scheme requires a fixed storage size per group, regardless of the number of entities. Indeed, each entity stores only two elements from $\mathbb{G}_1$ and $\mathbb{G}_2$. The proxy has to store only one element of $\mathbb{G}_T$ (244 bytes) for every group of entities.

Table 6.4: Performance Analysis. Required Storage Capacity

| | | **Size of secret elements (bytes)** |
|---|---|---|
| *CP-ABE* | Entity | $128 * N_i + 244$ |
| *Our Approach* | Each Entity | 128 |
| | Proxy | $244 * N_g$ |

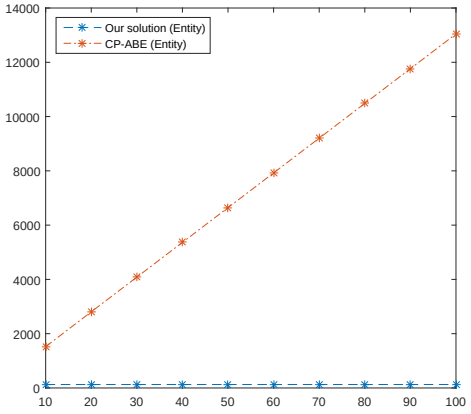



Figure 6.4: Size of secret elements for both the proxy and an entity in both of our approach and original CP-ABE

Figure 6.4 shows the variation of the size of an entity's and proxy's secret elements in both of our approach and original CP-ABE with respect to the number of attributes in secret key (The number of entities in the group). In our approach, the required storage size for an entity is independent from the number of entities.

## 6.4 Application cases

- **Location access control:** in this example, we consider a security system that controls location access for authorized persons. In order to provide a high security level, the control system authorizes the access only if the user holds, simultaneously, $k$ nodes of his authenticated devices (e.g. phone, smart bracelet, and smart glasses). In this way, even if one device is stolen, the thief will not be allowed to access the location. The system is composed of a proxy that is charged of

interrogating user's devices and a verifier entity which receives the proof and decides to authorize the access or not. An illustration is provided in Figure 6.5.

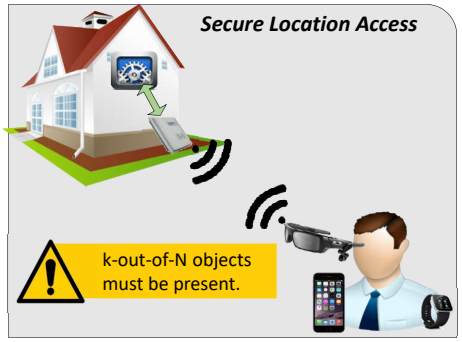



Figure 6.5: Illustration of the location access control system

When the user asks for access, the request is directed to the verifier and the latter starts the proof. The goal is to prove if the user is the right person by verifying the presence of $k$ nodes of his group of authenticated devices. Therefore, the verifier encrypts a random message and sends it to the proxy. In its turn, the proxy requests the group of devices and sends to each one of them its related part of the ciphertext. If $k$ or more nodes respond, the message can be decrypted. By sending the decrypted message to the verifier, the latter considers that the user is authorized and the access is allowed. Otherwise (timeout release), the user will not have access to the location.

- **Mobile NFC payment:** NFC payment emerged as a technology that allows users to issue payment transactions using their NFC-enabled mobiles. The use of mobile devices helped to enhance the security level and make transactions more secure than NFC cards (require certification for applications, use of authentication for users, etc.). However, if the phone gets stolen, payments can be issued by an unauthorized person. In fact, some clients may deactivate protection control features because they are often centered on user's interaction (e.g. entering the PIN code to issue the transaction). In this example we provide an application of our grouping proposal scheme to secure mobile NFC payments. The application is based on authenticating user through the simultaneous presence of his devices without requiring his interaction. Thus, when the user requests a NFC payment transaction, the mobile encrypt a random message and sends it to each user's device its related part of the ciphertext (in this example, the mobile is considered as proxy and verifier). the secret can be decrypted only if at least $k$ devices respond before the release of timer. If so (secret decrypted), the mobile continue the transaction.

## 6.5 Conclusion

In this chapter we have presented a solution to implement a threshold yoking/grouping proof scheme using Ciphertext- Policy Attribute-Based Encryption. Our approach provides more resistance against attacks by encrypting a secret in a such way that it can be recovered only if at least

k entities are present at the same time. In addition to standard yoking/group- ing proof proprieties, our solution introduces other features, such as the variability of the threshold k and the importance of user's entities.

# Chapter 7

# Conclusions and future directions

The Internet of Things must be designed for easy use, masking the underlying technological complexity, and for peaceful manipulation preventing security attacks. In IoT any object is potentially connected to the Internet and able to communicate with other objects. This creates new risks related to the confidentiality, authenticity and integrity of data that is sensed, collected and exchanged between objects. The privacy of individuals must be protected to avoid unauthorized identification and localization. As objects become more autonomous and more intelligent, problems relating to privacy are multiplied. Moreover, the strong integration of IoT with the physical world increases control over this world, but also makes it vulnerable to the potentially hazardous actions of the objects.

## 7.1 Conclusions

In this thesis, we were interested in the problem of access control given the sensitivity of targeted IoT applications and the eventual great damage in case of hazardous access to data or services. We identified Attribute based Encryption as a primary basis for the development of fine-grained access control schemes. Indeed, ABE allows sharing encrypted data and provides high expressiveness of access policies thanks to the attributes that are used to define both users' keys and access policies.

Nevertheless, many locks makes ABE schemes' implementation in the context of IoT too complex. In this thesis we tackled two obstacles: resource constraints of IoT objects and the absence of key/attribute revocation mecanisms that scale to the huge size of IoT.

We have proposed two distributed and collaborative versions for both CP-ABE and KP-ABE schemes in heterogeneous IoT environments. Our solutions allow to offload the computation overhead from resource-constrained devices to more powerful trusted neighborhood assistant nodes and a remote server. We showed through simulations and experiments that our solution is distinctly more efficient than original CP-ABE and KP-ABE.

We have also proposed an efficient batch-based attribute revocation mechanism for CP-ABE. Batch revocation optimizes the overhead due to re-keying and attribute reassignment. It is applicable in the case where revocation schedule is known a priori or when it tolerates some delay

before becoming effective. We proposed therefore different variants depending on the applications' requirements and revocation schedule assumptions.

We were also interested in the problem of grouping proof which means providing the proof that a group of objects is simultaneously present (at the same time in the same location). This service may have many applications aiming to strengthen the security level through requiring the simultaneous presence of a set of objects: wireless payment, access to sensitive buildings, etc. We have proposed a CP-ABE based solution allowing to prove the presence of a threshold of $k$ objects among $N$. We have also introduced the concept of entity importance to give to the presence of a particular object more importance in the grouping proof.

## 7.2 Open Issues and Future directions

Hereafter we shed some light on future directions and open issues relating to securing the Internet of Things:

- The pervasive nature of IoT raises legitimate questions about the privacy of persons, and how to cope with the heterogeneity of user and application requirements in terms of security services. This requires the development of adaptive, context-aware and user-centric security solutions. This diversity in terms of security requirements can be addressed via an adaptive, context-aware management of security policies.

- The perceptual and actional capabilities of objects and the possibilities of geo-localization raise a number of concerns for the privacy of users. Without special measures, objects could disclose sensitive information about individuals, such as itineraries, medical records, energy consumption, products consumed, clothing preferences, etc. Methods must be developed to allow objects to provide a privacy-aware data processing[29]. Existing data anonymization techniques require high capacities in terms of computing power, memory and bandwidth[29]. The resource constraints of objects and their networks make them difficult to use. New developments of inexpensive anonymization techniques are needed.

- The ubiquity of communicating objects facilitates the sharing of contents, entertainment, and even resources. The ubiquity of networks coupled with a highly dynamic, seamless mobility of communicating objects will lead to more sharing and encourage the emergence of a new sharing vector through nomadism. We believe that this sharing, driven by mobility and nomadism, will be a prime target for security attacks that in their turn will be encouraged and facilitated by the ubiquity of objects that are potential gateways to private networks and data. It would then be necessary to develop effective solutions for peaceful secure sharing through adequate access control mechanisms supporting mobility. We should consider the ubiquity of communicating objects and their mobility (transfer of security context) to design peer-to-peer sharing systems, which are secure, efficient and equitable, while supporting mobility

- Privacy requires fine-grained access control to avoid access to private information by third parties. However, what should be protected from disclosure depends heavily on user's and/or system's context. Moreover, the access policy may evolve following a change in the user's context. What must remain confidential under some circumstances, may be a vital input for a third party for user's safety and security. For instance, a person may deny access to her/his location to preserve her/his privacy. If it come that the same person falls in a isolated location and needs help, activation to her/his location may be vital. Therefore, we believe that access control in IoT must be extrapolated to become Activity Control given the capabilities provided by smart objects to sense their context.

Finally, we hope that some concepts and ideas introduced in this thesis will pave the way for a reliable and safe deployement of Internet of Things applications.

# Chapter 8

# Appendix

## 8.1 Appendix A: Proof $ANT \approx 1/\mu * \Delta t$

Let $T$ be a random variable that represents the requested attribute validity period (see figure example). $T$ follows an exponential distribution with parameter $\mu$ and $f(t) = \mu \cdot e^{-\mu t}$ is its probability density function.

Let $\Delta t$ be a fixed value representing time slot duration.

Let $N$ be the discrete random variable representing the number of time slots in the validity period $T$. This random variable determines the number of generated secret key parts (see formula scheme).

We cannot determine the value of $N$ with respect to $T$ because there are different cases for the same value of $T$ depending on the position of the start date of the validity period. Sometimes we have $N = \lceil \frac{T}{\Delta t} \rceil$. In other cases we have $N = \lfloor \frac{T}{\Delta t} \rfloor$.

In general, we have:

$$\left\lfloor \frac{T}{\Delta t} \right\rfloor \leq N \leq \left\lceil \frac{T}{\Delta t} \right\rceil \tag{8.1}$$

The average number of time slots per validity period (ANT) is equal to $\mathbb{E}(N)$.

Let $N_1$ and $N_2$ two random variables defined as follows:

$$N_1 = \left\lfloor \frac{T}{\Delta t} \right\rfloor ;\ N_2 = \left\lceil \frac{T}{\Delta t} \right\rceil$$

We have: $N_1 \leq N \leq N_2$ and we deduce that: $\mathbb{E}(N_1) \leq \mathbb{E}(N) \leq \mathbb{E}(N_2)$.

$$\mathbb{E}(N_1) = \sum_{n=1}^{+\infty} n \cdot P_1(n);\ (where : P_1(n) = P(N_1 = n)) \tag{8.2}$$

$$\begin{aligned} P_1(n) &= P(N_1 = n) \\ &= P(n.\Delta t \leq T < (n+1).\Delta t) \\ &= \int_{n.\Delta t}^{(n+1).\Delta t} f(t)dt \\ &= \int_{n.\Delta t}^{(n+1).\Delta t} \mu e^{-\mu t} dt \\ &= e^{-\mu n \Delta t}(1 - e^{\mu \Delta t}) \end{aligned} \tag{8.3}$$

From results 8.2 and 8.3 we have:

$$\begin{aligned} \mathbb{E}(N_1) &= \sum_{n=1}^{+\infty} n.e^{-\mu n \Delta t}(1 - e^{-\mu \Delta t}) \\ &= (1 - e^{-\mu \Delta t}) \cdot \sum_{n=1}^{+\infty} n.e^{-\mu n \Delta t} \\ &= (1 - e^{-\mu \Delta t}) \cdot \frac{e^{-\mu \Delta t}}{(1 - e^{-\mu \Delta t})^2} \\ &= \frac{e^{-\mu \Delta t}}{(1 - e^{-\mu \Delta t})} \\ &= \frac{1}{e^{\mu \Delta t} - 1} \\ &= \frac{e^{\mu \Delta t}}{e^{\mu \Delta t} - 1} - 1 \end{aligned} \tag{8.4}$$

similarly, we have:

$$\mathbb{E}(N_2) = \sum_{n=1}^{+\infty} n \cdot P_2(n);\ (where : P_2(n) = P(N_2 = n)) \tag{8.5}$$

$$\begin{aligned} P_2(n) &= P(N_2 = n) \\ &= P((n-1).\Delta t < T \leq n.\Delta t) \\ &= \int_{(n-1).\Delta t}^{n.\Delta t} f(t)dt \\ &= \int_{(n-1).\Delta t}^{n.\Delta t} \mu e^{-\mu t} dt \\ &= e^{-\mu n \Delta t}(e^{\mu \Delta t} - 1) \end{aligned} \tag{8.6}$$

From results 8.5 and 8.6 we have:

$$\begin{aligned}\mathbb{E}(N_2) &= \sum_{n=1}^{+\infty} n.e^{-\mu n\Delta t}(e^{\mu\Delta t}-1) \\ &= (e^{\mu\Delta t}-1)\cdot\sum_{n=1}^{+\infty} n.e^{-\mu n\Delta t} \\ &= (e^{\mu\Delta t}-1)\cdot\frac{e^{-\mu\Delta t}}{(1-e^{-\mu\Delta t})^2} \\ &= \frac{1}{1-e^{-\mu\Delta t}} \\ &= \frac{e^{\mu\Delta t}}{e^{\mu\Delta t}-1}\end{aligned} \tag{8.7}$$

Let set

$$\alpha = \frac{e^{\mu\Delta t}}{e^{\mu\Delta t}-1}$$

From 8.4 and 8.7:

$$\alpha - 1 \leq \mathbb{E}(N) \leq \alpha \tag{8.8}$$

Near $\mu\Delta t = 0$ (time slot duration very small compared to the average duration of a validity period), we have:

$$\alpha = \frac{1}{\mu\Delta t} + \frac{1}{2} + o(1) \tag{8.9}$$

$$\alpha - 1 = \frac{1}{\mu\Delta t} - \frac{1}{2} + o(1) \tag{8.10}$$

From formulas 8.9 and 8.10,

$$\frac{1}{\mu\Delta t} - \frac{1}{2} + o(1) \leq \mathbb{E}(N) \leq \frac{1}{\mu\Delta t} + \frac{1}{2} + o(1) \tag{8.11}$$

And from formula 8.11 we can approximate $\mathbb{E}(N)$ by $\frac{1}{\mu\Delta t}$

## 8.2 Appendix B: Proof $AWT \approx \Delta t/2$

We begin by naming time slots start dates by: $Tb$, therefore, the $i^{th}$ time slot can be represented by the interval $]Tb_i, Tb_{i+1}]$.

For a fixed time slot $i$, Let consider $X$ be a random variable representing the start date of an event compared to the start date of the $i^{th}$ time slot. We suppose that $X$ occurs during the $i^{th}$ time slot.

We are interested in the delay between the event date and the start date of the next time slot. Let $D$ be a random variable representing that delay. We have $D = \Delta t - X$. Where $\Delta t$ represents the time slot duration, $\Delta t = Tb_{i+1} - Tb_i$.

Let $E$ be a random variable representing the event start date independently of the start date of the time slot.

Then, We have:

$$\begin{aligned} &\forall B \in \mathcal{B}(\mathbb{R}); \\ &P(X \in B) = P(E - Tb_i \in B | E \in ]Tb_i, Tb_{i+1}]) \\ &\qquad = P(E - Tb_i \in B | E > Tb_i, E \leq Tb_{i+1}) \end{aligned} \tag{8.12}$$

Let $t$ and $Z$ be two random variables representing the time of the previous event and the period between the previous and the current events respectively. $Z$ follows an exponential distribution with parameter $\lambda$.

$$Z = E - t \tag{8.13}$$

Let set $s = Tb_i - t$, we deduce $Z - s = E - Tb_i$

We get:

$$\begin{aligned} &\forall B \in \mathcal{B}(\mathbb{R}); \\ &P(X \in B) = P(Z - s \in B | Z - s > 0, Z - s \leq \Delta t) \end{aligned} \tag{8.14}$$

Let $Y$ be a random variable defined as follows:

$$\begin{aligned} &\forall B \in \mathcal{B}(\mathbb{R}); \\ &P(Y \in B) = P(Z - s \in B | Z - s > 0) \end{aligned} \tag{8.15}$$

The random variable $Y$ follows a exponential distribution with parameter $\lambda$. It represents the period between the start date of the $i^th$ time slot and the event of the Poisson process.

From 8.14 and 8.15:

$$\begin{aligned} &\forall B \in \mathcal{B}(\mathbb{R}); \\ &P(X \in B) = P(Y \in B | Y \leq \Delta t) \end{aligned} \tag{8.16}$$

Now, let determine the cumulative distribution function of $X$

**Case 1:** $x \geq \Delta t$

$$\begin{aligned} F_X(x) &= P(X \leq x) \\ &= P(Y \leq x | Y \leq \Delta t) \\ &= \frac{P(Y \leq x; Y \leq \Delta t)}{P(Y \leq \Delta t)} \\ &= \frac{P(Y \leq \Delta t)}{P(Y \leq \Delta t)} \\ &= 1 \end{aligned} \tag{8.17}$$

**Case 2:** $x \leq 0$

$$\begin{aligned} F_X(x) &= P(X \leq x) \\ &= P(Y \leq x | Y \leq \Delta t) \\ &= 0 \end{aligned} \tag{8.18}$$

**Case 3:** $0 \leq x < \Delta t$

$$\begin{aligned} F_X(x) &= P(X \leq x) \\ &= P(Y \leq x | Y \leq \Delta t) \\ &= \frac{P(Y \leq x}{P(Y \leq \Delta t)} \\ &= \frac{\int_0^x f_Y(t)dt}{\int_0^{\Delta t} f_Y(t)dt} \\ &= \frac{\int_0^x \lambda e^{-\lambda t}dt}{\int_0^{\Delta t} \lambda e^{-\lambda t}dt} \\ &= \frac{1 - e^{-\lambda x}}{1 - e^{-\lambda \Delta t}} \end{aligned} \tag{8.19}$$

By differentiation, we get the probability density function of the random variable $X$.

$$f_X(x) = \begin{cases} \dfrac{\lambda e^{-\lambda x}}{1 - e^{-\lambda \Delta t}} & \text{if} : 0 < x < \Delta t \\ 0 \quad else \end{cases} \tag{8.20}$$

The average waiting time can be given by:

$$\begin{aligned} \mathbb{E}(X) &= \int_{-\infty}^{+\infty} x f_X(x)dx \\ &= \int_0^{\Delta t} x \frac{\lambda e^{-\lambda x}}{1 - e^{-\lambda \Delta t}}dx \\ &= \frac{-e^{-\lambda \Delta t}(\Delta t + \frac{1}{\lambda}) + \frac{1}{\lambda}}{1 - e^{-\lambda \Delta t}} \end{aligned} \tag{8.21}$$

From formula 8.21, the average waiting time $AWT$ reach its highest value ($\Delta t/2$) when $\lambda$ is very small (near to 0).

## 8.3 Appendix C: A simple definition of the our hash-function

In this annexe, we provide a possible definition of the one way hash function used in this paper. This definition is based on a standard hash function used in [11].

Let $H$ be a standard one way hash function defined as follows:

$$\begin{array}{rcccc} H & : & \{0,1\}^* & \longrightarrow & \mathbb{G}_0 \\ & & x & \longmapsto & H(x) \end{array} \tag{8.22}$$

This definition complies with the one of a standard one way hash functions (for example: SHA-1, MD5, ...).

Now, let us provide a simple definition for our one way hash function $H'$ that we introduced in Section 5.2.1.

Let $\mathbb{A}$ be the set of all attributes used in the system. $\mathbb{T}$ is the set of all time slot identifiers (we have $\mathbb{T} \subset \mathbb{N}$). A simple definition of $H'$ could be:

$$\begin{array}{rcccc} H' & : & \mathbb{A} \times \mathbb{T} & \longrightarrow & \mathbb{G}_0 \\ & & (x,i) & \longmapsto & H'(x,i) = H(x \| ' - ' \| i) \end{array} \tag{8.23}$$

Where $\|$ operator is the concatenation operator, and $'-'$ is a separator character that is not used to describe attributes.

**Proposition 1.** *The probability of collision existence in the function $H'$ is the same as with the function $H$.*

***Proof:*** Let $att_i, att_j \in \mathbb{A}$ be two attributes that are not necessarily different. Let $k, l \in \mathbb{T}$ be two time slot identifiers that are not necessarily different.

We have:

$$\begin{aligned} (att_i, k) \neq (att_j, l) &\Leftrightarrow (att_i \neq att_j) \; or \; (k \neq l) \\ &\Leftrightarrow att_i \| ' - ' \| k \neq att_j \| ' - ' \| l \end{aligned} \tag{8.24}$$

Let suppose that : $(att_i, k) \neq (att_j, l)$. From 8.23 and 8.24, we deduce:

$$H'(att_i, k) = H'(att_j, l) \Rightarrow H(att_i \| ' - ' \| k) = H(att_j \| ' - ' \| l) \tag{8.25}$$

We proved that if our one way function $H'$ leads to a collision then, the standard one way hash function $H$ (used in [11]) does too.

# Bibliography

[1] For powertop saving power on ia isn't everything. it is the only thing! `https://01.org/powertop`. Accessed: 2016-06-16.

[2] The gnu multiple precision arithmetic library. `https://gmplib.org/`. Accessed: 2016-04-09.

[3] Nfc forum. http://www.nfc-forum.org.

[4] Pbc library: The pairing-based cryptography library. `https://crypto.stanford.edu/pbc/`. Accessed: 2016-04-09.

[5] B. Anggorojati, P.N. Mahalle, N.R. Prasad, and R. Prasad. Capability-based access control delegation model on the federated iot network. In *Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on*, pages 604–608, Sept 2012.

[6] Muhammad Asim, Milan Petkovic, and Tanya Ignatenko. Attribute-based encryption with encryption and decryption outsourcing. 2014.

[7] Nuttapong Attrapadung, Javier Herranz, Fabien Laguillaumie, Benoît Libert, Elie De Panafieu, and Carla Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical Computer Science*, 422:15–38, 2012.

[8] Nuttapong Attrapadung, Benoît Libert, and Elie De Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *International Workshop on Public Key Cryptography*, pages 90–108. Springer, 2011.

[9] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[10] M. Bazzani, D. Conzon, A. Scalera, M. A. Spirito, and C. I. Trainito. Enabling the iot paradigm in e-health solutions through the virtus middleware. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1954–1959, June 2012.

[11] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 321–334, May 2007.

[12] Giuseppe Bianchi, Angelo T. Capossele, Chiara Petrioli, and Dora Spenza. Agree: Exploiting energy harvesting to support data-centric access control in wsns. *Ad Hoc Netw.*, 11(8):2625–2636, November 2013.

[13] Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart. *Advances in Elliptic Curve Cryptography*. London Mathematical Society Lecture Note Series (No. 317), April 2005.

[14] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *In EUROCRYPT*, pages 127–144. Springer-Verlag, 1998.

[15] Leonid Bolotnyy and Gabriel Robins. Generalized "yoking-proofs" for a group of rfid tags. IEEE, 4 2007.

[16] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Berlin: Springer-Verlag, 2005.

[17] Stefan Brands and David Chaum. Distance-bounding protocols. In *Advances in Cryptology-EUROCRYPT'93*, pages 344–359. Springer, 1993.

[18] Cheng Chen, Zhenfeng Zhang, and Dengguo Feng. Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In Xavier Boyen and Xiaofeng Chen, editors, *Provable Security*, volume 6980 of *Lecture Notes in Computer Science*, pages 84–101. Springer Berlin Heidelberg, 2011.

[19] Yong Cheng, Jiangchun Ren, Zhiying Wang, Songzhu Mei, and Jie Zhou. Attributes union in cp-abe algorithm for large universe cryptographic access control. In *Cloud and Green Computing (CGC), 2012 Second International Conference on*, pages 180–186, Nov 2012.

[20] Yong Cheng, Zhi-ying Wang, Jun Ma, Jiang-jiang Wu, Song-zhu Mei, and Jiang-chun Ren. Efficient revocation in ciphertext-policy attribute-based encryption based cryptographic cloud storage. *Journal of Zhejiang University SCIENCE C*, 14(2):85–97, 2012.

[21] Jung-Sik Cho, Sang-Soo Yeo, Suchul Hwang, Sang-Yong Rhee, and Sung Kwon Kim. Enhanced yoking proof protocols for rfid tags and tag groups. In *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, pages 1591–1596, March 2008.

[22] Alexei Czeskis, Karl Koscher, Joshua R Smith, and Tadayoshi Kohno. Rfids and secret handshakes: defending against ghost-and-leech attacks and unauthorized reads with context-aware communications. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 479–490. ACM, 2008.

[23] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Fully secure online/offline predicate and attribute-based encryption. In Javier Lopez and Yongdong Wu, editors, *Information Security Practice and Experience*, volume 9065 of *Lecture Notes in Computer Science*, pages 331–345. Springer International Publishing, 2015.

[24] JosÃ© M. de Fuentes, Pedro Peris-Lopez, Juan E. Tapiador, and Sergio Pastrana. Probabilistic yoking proofs for large scale iot systems. *Ad Hoc Networks*, 32:43 – 52, 2015. Internet of Things security and privacy: design methods and optimization.

[25] J. Deng, C. Hartung, R. Han, and S. Mishra. A practical study of transitory master key establishment for wireless sensor networks. In *IEEE Intl Conf on Security and Privacy for Emerging Areas in Comm. Networks*, September 2005.

[26] Dang Nguyen Duc and Kwangjo Kim. Grouping-proof protocol for rfid tags: Security definition and scalable construction. 2009.

[27] Tom J. Kamierski (Ed.) and Steve Beeby (Ed.). *Energy Harvesting Systems: Principles, Modeling and Applications*. Springer, 2010.

[28] Keita Emura, Atsuko Miyaji, Akito Nomura, Kazumasa Omote, and Masakazu Soshi. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In *Information Security Practice and Experience*, pages 13–23. Springer, 2009.

[29] Ovidiu Vermesan et al. Internet of things strategic research roadmap, 2011. Cluster of European Research Projects on the Internet of Things.

[30] ETSI. Machine-to-machine communications, 2011. WMC, Barcelona.

[31] European Telecommunications Standards Institute (ETSI). Machine- to- machine communications (m2m) : Functional architecture, 2011. Draft ETSI TS 102 690 V0.10.3.

[32] S. Galbraith, F. Hess, and F. Vercauteren. Aspects of pairing inversion. *Information Theory, IEEE Transactions on*, 54(12):5719–5728, 2008.

[33] Aijun Ge, Rui Zhang, Cheng Chen, Chuangui Ma, and Zhenfeng Zhang. Threshold ciphertext policy attribute-based encryption with constant size ciphertexts. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *Information Security and Privacy*, volume 7372 of *Lecture Notes in Computer Science*, pages 336–349. Springer Berlin Heidelberg, 2012.

[34] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM.

[35] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. In *Proceedings of the 20th USENIX conference on Security*, pages 34–34. USENIX Association, 2011.

[36] Fuchun Guo, Yi Mu, Willy Susilo, Duncan Wong, and Vijay Varadharajan. Cp-abe with constant-size keys for lightweight devices. *Information Forensics and Security, IEEE Transactions on*, 9(5):763–771, 2014.

[37] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. A capability-based security approach to manage access control in the internet of things. *Mathematical and Computer Modelling*, 58(5-6), 2013.

[38] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4):660–670, Oct 2002.

[39] Javier Herranz, Fabien Laguillaumie, and Carla Ràfols. Constant size ciphertexts in threshold attribute-based encryption. In PhongQ. Nguyen and David Pointcheval, editors, *Public Key Cryptography - PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 19–34. Springer Berlin Heidelberg, 2010.

[40] Susan Hohenberger and Brent Waters. Attribute-based encryption with fast decryption. In *Public-Key Cryptography–PKC 2013*, pages 162–179. Springer, 2013.

[41] Susan Hohenberger and Brent Waters. Online/offline attribute-based encryption. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 293–310. Springer Berlin Heidelberg, 2014.

[42] Luan Ibraimi, Qiang Tang, Pieter Hartel, and Willem Jonker. Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In *International Conference on Information Security Practice and Experience*, pages 1–12. Springer, 2009.

[43] Sonia Jahid and Nikita Borisov. Piratte: Proxy-based immediate revocation of attribute-based encryption. *arXiv preprint arXiv:1208.4877*, 2012.

[44] Jiong Jin, Jayavardhana Gubbi, Slaven Marusic, and Marimuthu Palaniswami. An information framework for creating a smart city through internet of things. *IEEE Internet of Things Journal*, 1(2):112–121, 2014.

[45] Yu Jin, Chuan Tian, Heng He, and Fan Wang. A secure and lightweight data access control scheme for mobile cloud computing. In *Big Data and Cloud Computing (BDCloud), 2015 IEEE Fifth International Conference on*, pages 172–179, Aug 2015.

[46] A. Juels. "yoking-proofs" for rfid tags. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 138–143, March 2004.

[47] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 568–588. Springer, 2011.

[48] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang. Securely outsourcing attribute-based encryption with checkability. *IEEE Transactions on Parallel and Distributed Systems*, 25(8):2201–2210, Aug 2014.

[49] Jing Liu, Yang Xiao, and C.L. Philip Chen. Internet of things' authentication and access control. *Int. J. Secur. Netw.*, 7(4):228–241, 2012.

[50] Di Ma, Navrati Saxena, Tuo Xiang, and Yan Zhu. Location-aware and safer cards: enhancing rfid security and privacy via location sensing. *Dependable and Secure Computing, IEEE Transactions on*, 10(2):57–69, 2013.

[51] P.N. Mahalle, P.A. Thakre, N.R. Prasad, and R. Prasad. A fuzzy approach to trust based access control in internet of things. In *VITAE*, pages 1–5, 2013.

[52] Robert McMillan. Power grid is found susceptible to cyberattack, 2009. http://www.computerworld.com/s/article/9130178/Power_grid_is_found_susceptible_to_cyberattack.

[53] Disruptive Civil Technologies National Intelligence Council. Six technologies with potential impacts on us interests out to 2025, 2008. Conference Report CR 2008 07.

[54] Neeli Rashmi Prasad Parikshit N. Mahalle, Bayu Anggorojati and Ramjee Prasad. Identity authentication and capability based access control (iacac) for the internet of things. *Journal of Cyber Security and Mobility*, 1(4):309–348, 2013.

[55] Piramuthu. On existence proofs for multiple rfid tags. In *Pervasive Services. the 2006 ACS/IEEE International Conference on*, pages 317 – 320, June 2006.

[56] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 99–112, New York, NY, USA, 2006. ACM.

[57] Tim Polk and Sean Turner. Security challenges for the internet of things. In *Workshop on Interconnecting Smart Objects with the Internet*, March 2011.

[58] Jie REN, Qiang WANG, and Jun-wei ZHAI. Study on the application of the technology of iot in military logistics [j]. *Logistics Sci-Tech*, 11:029, 2011.

[59] J. "Saito and K." Sakurai. Grouping proof for rfid tags. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, pages 621 – 624, March 2005.

[60] Luis Sanchez, Luis Muñoz, Jose Antonio Galache, Pablo Sotres, Juan R Santana, Veronica Gutierrez, Rajiv Ramdhany, Alex Gluhak, Srdjan Krco, Evangelos Theodoridis, et al. Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61:217–238, 2014.

[61] Reijo M. Savola, Habtamu Abie, and Markus Sihvonen. Towards metrics-driven adaptive security management in e-health iot applications. In *Proceedings of the 7th International Conference on Body Area Networks*, BodyNets '12, pages 276–281, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[62] Nitesh Saxena and Jonathan Voris. Still and silent: motion detection for enhanced rfid security and privacy without changing the usage model. In *Radio Frequency Identification: Security and Privacy Issues*, pages 2–21. Springer, 2010.

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

[64] T. Tsao, R. Alexander, M. Dohler, V. Daza, and A. Lozano. A security framework for routing over low power and lossy networks. IETF Draft, draft-ietf-roll-security-framework-06, June 2011.

[65] International Telecommunication Union. The internet of things. Report, November 2005.

[66] Changji Wang and Jianfa Luo. An efficient key-policy attribute-based encryption scheme with constant ciphertext length. *Mathematical Problems in Engineering*, 2013, 2013.

[67] Guojun Wang, Qin Liu, and Jie Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 735–737, New York, NY, USA, 2010. ACM.

[68] Guojun Wang, Qin Liu, Jie Wu, and Minyi Guo. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Computers & Security*, 30(5):320 – 331, 2011. Advances in network and system security.

[69] Y. Wei, J. Wang, and J. Wang. A delay/disruption tolerant routing algorithm for iot in harsh environment. In *2013 6th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pages 143–146, Nov 2013.

[70] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, and JP. Vasseur. Rpl: Ipv6 routing protocol for low power and lossy networks. IETF Draft, draft-ietf-roll-rpl-19, March 2011.

[71] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *IEEE Network*, 20(3):41–47, May-June 2006.

[72] Zhiqian Xu and K.M. Martin. Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 844–849, June 2012.

[73] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Attribute based data sharing with attribute revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 261–270, New York, NY, USA, 2010. ACM.

[74] Miao Yun and Bu Yuxin. Research on the architecture and key technology of internet of things (iot) applied on smart grid. In *Advances in Energy Engineering (ICAEE), 2010 International Conference on*, pages 69–72. IEEE, 2010.

[75] Zhibin Zhou and Dijiang Huang. Efficient and secure data storage operations for mobile cloud computing. In *Proceedings of the 8th International Conference on Network and Service Management*, pages 37–45. International Federation for Information Processing, 2012.