# Cloud computing security

Youcef Imine

▶ **To cite this version:**

Youcef Imine. Cloud computing security. Cryptography and Security [cs.CR]. Université de Technologie de Compiègne, 2019. English. NNT : 2019COMP2520 . tel-02882540

**HAL Id: tel-02882540**
**https://theses.hal.science/tel-02882540**

Submitted on 26 Jun 2020

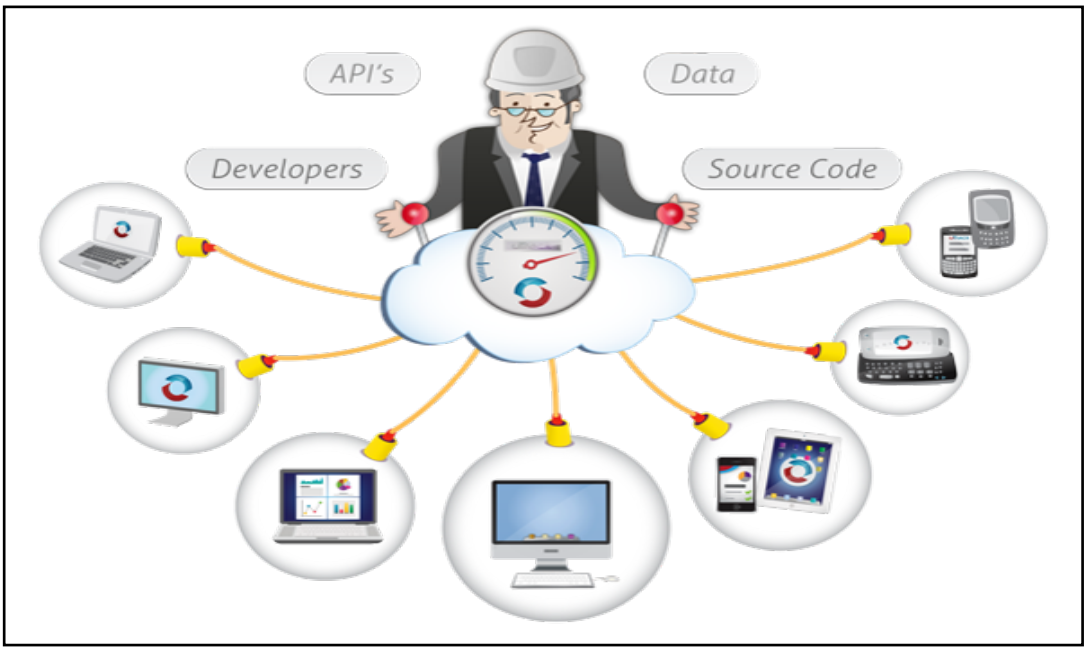
**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Par **Youcef IMINE**

# *Cloud computing security*

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




Soutenue le 18 novembre 2019
**Spécialité :** Informatique et Sciences et Technologies de l'Information et des Systèmes : Unité de recherche Heudyasic (UMR-7253)
D2520

# Cloud computing security

Youcef Imine

Spécialité :
Informatique et Sciences et Technologies de l'Information et des Systèmes

---

Thèse soutenue le 18 novembre 2019 devant le jury composé de :

**Président:**

*Walter SCHÖN*
Proffesseur
Université de Technologie de Compiègne

**Rapporteurs:**

*Ahmed Serhrouchni*
Professeur des université
Telecom ParisTech

*Francine Krief*
Professeur des universités
L'institut polytechnique de Bordeaux (Bordeaux INP)

**Examinateurs:**

*Isabelle Chrisment*
Professeur
TELECOM Nancy, Université de Lorraine

*Bijan Jabbari*
Professeur
Université de George Mason

*Romain Laborde*
MCF, HDR
Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier

**Directeurs de Thèse:**

*Abdelmadjid Bouabdallah* Professeur des universités
Université de Technologie de Compiègne

---

Université de Technologie de Compiègne

Laboratoire Heudiasyc UMR CNRS 7253

18 - 11 - 2019

# *Contents*

# *List of figures*

# *List of tables*

# *List of algorithms*

# *Abstract*

These last years, we are witnessing a real digital revolution of Internet where many innovative applications such as Internet of Things, autonomous cars, etc., have emerged. Consequently, adopting externalization technologies such as cloud and fog computing to handle this technological expansion seems to be an inevitable outcome. However, using the cloud or fog computing as a data repository opens many challenges in prospect.

This thesis addresses security issues in cloud and fog computing which is a major challenge that need to be appropriately overcomed. Indeed, adopting these technologies means that the users lose control over their own data, which exposes it to several security threats. Therefore, we first investigated the main security issues facing the adoption of cloud and fog computing technologies.

As one of the main challenges pointed in our investigation, access control is indeed a cornerstone of data security. An efficient access control mechanism must provide enforced and flexible access policies that ensure data protection, even from the service provider. Hence, we proposed a novel secure and efficient attribute-based access control scheme for cloud data-storage applications. Our solution ensures flexible and fine-grained access control and prevents security degradations. Moreover, it performs immediate users and attributes revocation without any key regeneration.

Authentication service in fog computing architecture is another issue that we have addressed in this thesis. Some traditional authentication schemes endure latency issues while others do not satisfy fog-computing requirements such as mutual authentication between end-devices and fog servers. Thus, we have proposed a new, secure and efficient authentication scheme that ensures mutual authentication at the edge of the network and remedies to fog servers' misbehaviors.

Finally, we tackled accountability and privacy-preserving challenges in information-sharing applications for which several proposals in the literature have treated privacy issues, but few of them have considered accountability service.

Therefore, we have proposed a novel accountable privacy-preserving solution for public information sharing in data externalization platforms. Externalization servers in our scheme authenticate any user in the system without violating its privacy. In case of misbehavior, our solution allows to trace malicious users thanks to an

authority.

**Keywords:** Cloud computing, fog computing, security, access control, revocation, authentication, privacy, accountability.

# *Résumé*

Ces dernières années, nous assistons à une immense révolution numérique de l'Internet où de nombreuses applications innovantes telles que l'Internet des objets, les voitures autonomes, etc., ont émergées. Par conséquent, l'adoption des technologies d'externalisations des données, telles que le cloud ou le fog computing, afin de gérer cette expansion technologique semble inévitable. Cependant, l'utilisation du cloud ou du fog computing en tant que plateforme d'externalisation pour le stockage ou le partage des données crée plusieurs défis scientifiques. En effet, externaliser ses données signifie que l'utilisateur perd le contrôle sur ces derniers. D'où, la sécurité des données devienne une préoccupation majeure qui doit être proprement traitée. C'est dans ce contexte que s'inscrivent les travaux de cette thèse dans laquelle nous avons déterminé dans un premier temps les principaux problèmes de sécurité liés à l'adoption du cloud et du fog computing.

Le contrôle d'accès aux données est l'un des défis majeurs que nous avons identifié. Un mécanisme de contrôle d'accès efficace doit permettre d'appliquer des politiques d'accès fiables, flexibles et qui garantissent la protection des données contre toute sorte d'accès non autorisé venant des utilisateurs ou du fournisseur de service. De ce fait, nous avons proposé une nouvelle solution de contrôle d'accès basée sur le chiffrement à base d'attributs pour les applications de stockage de données dans le cloud. Notre solution assure un contrôle d'accès souple et à grains fins. De plus, elle permet d'effectuer une révocation immédiate des utilisateurs et des attributs sans aucune mise à jour des clés de chiffrement fournies aux utilisateurs.

Le service d'authentification dans une architecture fog computing est un autre problème que nous avions abordé durant cette thèse. En effet, certains schémas traditionnels d'authentifications proposés dans la littérature sont confrontés à des problèmes de latence, tandis que d'autres ne sont pas conformes aux exigences du fog computing telles que l'authentification mutuelle entre les utilisateurs et les serveurs Fog. Ainsi, nous avons proposé un nouveau schéma d'authentification efficace, qui assure l'authentification mutuelle et qui est robuste contre les comportements malicieux des serveurs Fog.

Enfin, nous avons abordé le problème de traçabilité et de la protection de la vie privée dans le cadre des applications de partage d'informations publiques. Plusieurs

propositions dans la littérature ont traité les problèmes liés à la protection de la vie privée. Cependant, peu de solutions ont envisagé un service de traçabilité.

Par conséquent, nous avons proposé une nouvelle solution pour le partage d'informations publiques assurant le service de traçabilité tout en préservant les informations privées des utilisateurs. Avec notre solution, les serveurs d'externalisations authentifient les utilisateurs sans pouvoir obtenir des informations sur leurs vies privées. En cas de comportements malicieux, notre solution permet de tracer les utilisateurs malveillants grâce à une autorité.

**Mots clés:** Cloud computing, fog computing, contrôle d'accès, révocation, authentification, vie privée, traçabilité.

# *Remerciements*

Avant tout, je souhaite remercier très respectueusement Monsieur Abdelmadjid BOUABDALLAH de m'avoir accepté en premier lieu pour cette thèse et pour son encadrement professionnel, ses conseils, son esprit d'équipe et le soutien qu'il apporte à ses doctorants. Je tiens de même à remercier Monsieur Ahmed Lounis pour sa coopération, sa vision et son esprit critique qui étaient des éléments importants du succès de cette thèse.

Je remercie également Madame Francine KRIEF Professeur à l'Université de Bordeaux et Monsieur Ahmed SERHROUCHNI, Professeur à Télécom ParisTech, qui ont pris de leur temps pour rapporter mon mémoire.

Je remercie Madame Isabelle CHRISMENT, Professeur à Telecom Nancy et Messieurs Walter SCHÖN, professeur à l'Université de Technologie de Compiègne, Bijan JABBARI professeur à l'Université de George Mason et Romain LABORDE, MCF-HDR à l'Université de Paul Sabatier, qui ont bien voulu faire partie de mon jury de soutenance.

J'adresse les plus chaleureux remerciements à toute ma famille notamment à mes parents, mes grands-parents, ma tante LEBEKIA NAOUEL et mon oncle LEBEKIA ABDELATIF, pour leur soutien continu et leurs encouragements.

Enfin, je remercie mes amis et collègues du laboratoire Heudiasyc pour leurs accueil, l'aide qu'ils ont su m'apporter durant ma thèse et surtout pour la convivialité ainsi que la bonne ambiance qui régnait au laboratoire.

# *Publications*

- Youcef Imine, Ahmed Lounis, Abdelmadjid Bouabdallah: Revocable attribute-based access control in mutli-autority systems. J. Network and Computer Applications 122: 61-76 (2018)
- Youcef Imine, Ahmed Lounis, Abdelmadjid Bouabdallah: ABR: A new efficient attribute based revocation on access control system. IWCMC 2017: 735-740
- Youcef Imine, Ahmed Lounis, Abdelmadjid Bouabdallah: Immediate Attribute Revocation in Decentralized Attribute-Based Encryption Access Control. TrustCom/BigDataSE/ICESS 2017: 33-40
- Youcef Imine, Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, Ahmed Lounis: MASFOG: An Efficient Mutual Authentication Scheme for Fog Computing Architecture. TrustCom/BigDataSE 2018: 608-613
- Youcef Imine, Ahmed Lounis, Abdelmadjid Bouabdallah: An Efficient Accountable Privacy-Preserving Scheme for Public Information Sharing in Fog Computing. Globecom 2019.

# Introduction

## 1.1 Motivation

Cloud computing is a flexible and dynamic storage and execution environment that provides its users with on-demand computing resources via the Internet. Cloud computing offers many advantages thanks to the major evolution of virtualization techniques. Indeed, with the cloud model, it is possible to outsource data to remote servers on demand, usually by use and according to technical criteria such as power, storage space, bandwidth, etc. As a result, users can loan computational resources according to their needs whenever they wanted to, instead of paying equipment that might cost them a lot of money. In addition, this model provides mobility and flexibility since the access is possible from anywhere via Internet.

With the emergence of many applications such as internet of things (IoT), smart cities, autonomous cars, etc. Cloud computing places itself as an important player in data supply/demand equation that the world is about to face in the coming years. Indeed, According to Cisco [Cis, 2018], the Annual global IP traffic will reach 3.3 Zettabytes by 2021. Thus, adopting cloud computing as a solution to handle these considerable amounts of data seems to be an inevitable conclusion.

However, cloud computing has a centralized operating mode which can become problematic. In fact, when the processed data raises up, it will endure more latency and so a lack of efficiency, especially for real time applications. Consequently, a new paradigm named fog computing, has appeared recently to overcome these limitations. Fog paradigm aims to extend cloud services to the edge of the network while ensuring interaction with the cloud. Therefore, computation, communication, storage and control operations are performed closer to the end user by pooling network's local resources. Fog computing paradigm complements cloud computing since it remedies to low latency by managing local information in the edge of the network, while keeping the coordination and global analytics to the cloud.

Neverthless, the users need to outsource their data into external servers in both cases, which means that they lose control of their data to benefit from cloud or fog services. That is why security places it self as one of the major preoccupations that

need to be addressed while adopting these technologies. Indeed, besides traditional issues related to software security, data transfer security, virtual machine isolation, etc., threats related to access control breach; the loss of data confidentiality; the violation of users privacy may have a tragic impact on compagnies and users adopting cloud or fog computing as their main data repository. Therefore, how can we benefit from these technologies while keeping control of data on the users' hands?

## 1.2 Research topic

Given the pros and cons of sharing data in the cloud or fog computing, it is essential to propose robust security solutions, which: (1) ensure a high confidentiality of the outsourced data; (2) support users' mobility and the heterogeneity of the data-sharing environment (especially in fog computing); (3) satisfy the level of quality of service required by users.

To this end, we address in this thesis three main problems in the context of cloud and fog computing:

- **Access control in data sharing applications** is one of the principal counter measurement that need to be implemented to secure data externalization process. Generally, it refers to the mechanisms ensuring that only legitimate users can get access to the data. While addressing access control, researchers usually focus on the attribution of access rights, the supervision of the access, and most importantly the management of revocation situations, where the users lose some of their access credentials. The revocation can be an easy task when all users have access to a single data-sharing domain. However, it becomes more complicated as soon as the users need to access to multiple domains with different access rights, or when the frequency of leaving the system or acquiring new access rights raises.

  Thus, in the context of data sharing in cloud computing, how can we set up a reliable and robust access control model which provides an effective control to the data owner and ensures an immediate and efficient revocation mechanism?

- **Authentication in fog computing** is also seen as a crucial challenges especially in untrustworthy architectures as fog computing. In fact, fog computing is susceptible to several attacks. One can cite replay attack and Man in the middle (MITM) that aim to impersonate legitimate fog nodes or users. Therefore, developing a robust authentication mechanism that allows both the users and the fog nodes to verify the authenticity of each other and deal efficiently with malicious attacks is one of the most important challenges

to address. In addition, since fog computing paradigm aims to overcome cloud computing shortcomings in terms of latency and bandwidth saving, proposing authentication solutions that do not rely too much on the cloud or any central authentication servers is another challenge in this new architecture [Hu et al., 2017].

- **Accountability and privacy-preserving in information sharing application** are two main challenges to address when outsourcing data, given that externalization servers might expose users' personal information to leakage threat. Even if the leakage of personal information problem can be solved using cryptography, preserving privacy is not limited in exposing users' identities or some of their private information to the public. It also includes the detection of users' behavior pattern, activity tracking, interests and preference detection, etc. In fact, selling this kind of information to companies, interested in targeted advertising for example, may be much more useful for service providers than revealing users' identity. Therefore, given a network of communicating entities that share public information in any data externalization plateform, how can we preserve the privacy of communicating entities? In addition, how can we prevent malicious users from taking advantage of privacy-preserving feature and act maliciously without being able to trace them?

## 1.3 Our contributions

**Contribution 1: (Revocable Attribute-Based Access Control In Mutli-Autority Systems)**

Multi-authority attribute-based encryption is an encryption method which provides a distributed, flexible and fine-grained access control in untrustworthy environments. However, this method suffers from some shortcomings as revocation which is one of its major challenges. The revocation consists of banishing users from the system or some of their attributes to prevent them from getting access to the data. To overcome this limitations, we proposed a novel and efficient revocation solution for decentralized attribute-based scheme. Our solution ensures flexible and fine-grained access control and prevents security degradations. Moreover, it performs immediate users and attributes revocation without performing any key regeneration on the users' side. In addition, it provides collusion resistance and supports scalability.

**Contribution 2: (MASFOG: An Efficient Mutual Authentication Scheme For Fog Computing Architecture)**

Authentication is the entry point of any security system, which makes it an important security service. Traditional authentication schemes endure latency issues and some of them do not satisfy fog-computing requirements such as mutual authentication between end devices and fog servers. Therefore, we proposed a new efficient authentication scheme for fog computing architecture. Our scheme ensures mutual authentication and remedies to fog servers' misbehaviors. Moreover, fog servers need to hold only a couple of information to verify the authenticity of every user in the system. Thus, it provides a low overhead in terms of storage capacity.

**Contribution 3: (An Accountable Privacy-Preserving Scheme for Public Information Sharing systems)**

The emergence of data externalization technologies, as cloud and fog computing, has considerably eased the deploiement of public information-sharing applications. Yet, many concerns related to information security need to be addressed. While sharing information, privacy is without any doubt one of the major concerns for all users. Whereas, when security systems do not adopt accountability mechanisms, full anonymity may encourage users to act maliciously.

Consequently, we proposed a novel accountable privacy-preserving solution for public information sharing in data externalization platforms. Based on signatures, our scheme allows externalization servers to authenticate any user in the system without violating its privacy. In case of misbehavior, our scheme considers an authority that is able to trace any user in the system. Both, privacy-preserving and accountability services are ensured in a completely distributed manner, without a permanent resort to the authority.

## 1.4 Organization of the manuscript

The rest of this thesis is organized as follows. In chapter 2, we present cloud and fog computing architectures along with some backgrounds on security techniques. In section 3, we discuss the security issues and challenges in cloud and fog computing. In Chapter 4, we present our revocable attribute-based access control system. In chapter 5, we present our mutual authentication scheme for fog computing architectures. In chapter 6, we present our accountable privacy-Preserving scheme for public information sharing systems. Finally, we provide a conclusion in chapter 7 along with the main future work directions and open issues.

# *Cloud/Fog computing architectures and security techniques*

## 2.1 Introduction

With the huge technological evolution that the world is witnessing in several fields, the need to share data has phenomenally grown up. Many innovative applications such as Internet of Things, autonomous cars, etc., require data storage and sharing through service platforms, such as cloud, fog-computing or any external server, and even in a fully distributed manner. Thus, securing the exchange of information becomes an important challenge, particularly in data sensitive applications, where the data-sharing process is exposed to several threats [Singh and Chatterjee, 2017, Alaba et al., 2017]. Indeed, besides data leakage threats due to eavesdropping, hacking and even components compromise, data owners cannot totally trust the cloud and fog service providers. Consequently, they should apply efficient counter measurement to ensure data confidentiality in these untrustworthy environments.

In this chapter, we introduce cloud and fog computing technologies and their characteristics in section 2.2 and 2.3. Then, we present security services and some of its mechanisms in section 2.4. Finally, we conclude in section 2.5

## 2.2 Cloud computing

Cloud computing is an efficient computing model which has widely spread in these last years. This paradigm enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources, such as networks, storage, software, etc., and which can be provisioned and released in a flexible manner with minimal management effort [Mell and Grance, 2011].Consequently, the users can loan the resources according to their needs instead of paying equipment that might cost them a lot of money. In addition, this model provides mobility and flexibility since the access can be at any time and from anywhere using internet

technology. According to [Mell and Grance, 2011], cloud computing need to provide the following five essential characteristics:

- **On-demand self-service**: customers can provision any computing resources, such as server time and storage space, etc., whenever it is needed, in a flexible way without requiring human interaction.
- **Broad network access**: computing services are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- **Resources pooling**: the cloud computing is a multi-tenant model since computing resources are shared between multiple consumers. Therefore, these resources need to be assigned and reassigned dynamically according to consumer demand.
- **Rapid elasticity**: computing resources need to easily be acquired or released in order to rapidly scale and adapt with demand, which make them often appear unlimited and can be appropriated at any time, from the customers' point of view.
- **Measured service**: the cloud ensures transparency for both the provider and the consumers, by providing tools that enable cloud customers to control and supervise usage of their resources.

In cloud computing, we can distinguish two main models [Mell and Grance, 2011]:

1. Services model: in which we have three kind of services:
    - **Software as a service (SaaS)**: is the most basic form of cloud computing. It includes implementation of specific applications such as ERP, CRM, Google Docs, etc.
    - **Platform as a service (PaaS)**: is a cloud service model in which the cloud delivers a platform to the users from which they can develop, initialize and manage applications. We cite as examples: Google's App Engine, IBM BlueMix, and Apache's Stratos, etc.
    - **Infrastructure as a service (IaaS)**: is the lowest-level cloud service that can be provided to customers. In fact, with IaaS, pre-configured hardware resources are provided to users through a virtual interface. Unlike PaaS and SaaS, IaaS does not include applications or even an

operating system (implementing all of that is left up to the customer), it simply enables access to the infrastructure needed to power or support that software. One of the most Popular IaaS are Amazon EC2, IBM SoftLayer, and Google's Compute Engine (GCE).

2. Deployment model: cloud computing can also be seen according to its deployment model, where we can find:

   - **Public Cloud**: which is a type of cloud hosting, in which the cloud services are delivered over a network that is open for public usage.
   - **Private Cloud**: known also as internal cloud, and in which the platform for cloud computing is implemented on a cloud-based secure environment and it belongs to a particular corporation.
   - **Hybrid Cloud**: which is a type of cloud computing in which two or more cloud servers, i.e. private and public cloud are bound together, but remain individual entities.
   - **Community cloud**: a community cloud in computing is a collaborative effort in which infrastructure is shared between several organizations from a specific community with common concerns (security, compliance, jurisdiction, etc.).

## 2.3 Fog computing

Recently a new paradigm called fog computing has appeared, it aims to extend cloud services to the edge of the network while ensuring interaction with the cloud. [Michaela Iorga, 2018] has defined this new paradigm as "*a horizontal, physical or virtual resource paradigm that resides between smart end-devices and traditional cloud or data centers*".

The fog computing paradigm provides the following characteristics:

- **Contextual location awareness and low latency**: the origins of fog computing can be traced to early proposals supporting endpoints with rich services at the edge of the network, including applications with low latency requirements (e.g. gaming, video streaming, and augmented reality). Since fog nodes tend to sit very close to end users, data analysis and response to the users will be much quicker compared to a centralized cloud. Therefore, it meets the demand of real time interactions, especially for latency-sensitive or time sensitive applications.

- **Geographical distribution**: in sharp contrast to cloud computing, fog computing fits more with services and applications that demand a widely distributed deployment, due to its geographically distribution by design. For instance, compared to the cloud, fog computing will be more reliable in delivering high quality streaming services to moving vehicles, through proxies and access points positioned along highways and tracks.

- **Large scale and mobility support**: in addition to its ability to deal efficiently with large scale applications, it is essential for many fog applications to communicate directly with mobile devices, and therefore support mobility.

- **Heterogeneity**: fog nodes come in different form factors, and will be deployed in a wide variety of environments. In addition, end users' devices may also vary in terms of network communication protocols, capabilities, etc.

- **Interoperability and federation**: seamless support of certain services such as real-time video streaming services requires the cooperation of different providers. Hence, fog components must be able to inter-operate, and services must be federated across domains.

## 2.4 Fundamentals of security

According to [Dukes, 2015], the security of information systems has been defined as *"the protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to ensure confidentiality, integrity, and availability"*. From there we can distinguish three main security services, which are:

- **Confidentiality**: which means protecting information from disclosure to unauthorized parties.

- **Integrity**: which refers to protecting information from being modified by unauthorized parties.

- **Availability**: which means that the data should be available all the time and authorized parties should be able to access the information when needed.

In addition, we find other security services that might be important in some scenarios and applications, such as:

- **Authentication**: which refers to the verification of users' identities.

- **Access control**: which refers to providing protection against unauthorized use of resources.
- **Non-repudiation**: which refers to the non-denial of any action performed by any user in the system.
- **Privacy**: which refers to the protection of personal information.

### 2.4.1 Security mechanisms

Security mechanisms are the technical tools and techniques that are used to implement security services. A mechanism might operate by itself, or with others, to provide a particular service. In this section we focus on two security mechanisms that we present in what follows:

#### 2.4.1.1 Cryptography

Cryptography or the art of hiding messages has become nowadays a science on its own; it combines multiple disciplines such as mathematics, computer sciences and even physics to ensure the protection of the data by providing several security services such as authenticity, integrity, confidentiality and non-repudiation. There are two main cryptographic schemes:

**Symmetric cryptography**: the symmetric encryption is the oldest and the fastest-known technique in cryptography. It consists of combining a common secret key with a message to change its content in a particular way. As long as both sender and recipient know the secret key, they can securely exchange messages using this shared key.

Symmetric key ciphers are valuable because:

- It does not require a considerable computational cost to produce a strong key for the ciphers.
- Compared to the level of protection they afford, the keys tend to be much smaller.
- The encryption/decryption algorithms are relatively fast to process.

The major drawback of symmetric encryption is in exchanging the secret key because any exchange must retain the privacy of the key. This usually means that the secret key need to be transmitted in a secure channel; or must be encrypted with a different key which leads to a never-ending dependency on another key. In addition, a user

needs to define a new secret key for each communication established with another user. Therefore, he will store as many keys as the number of users with whom he established a communication link.

**Asymmetric cryptography**: known also as public key cryptography, is a cryptographic scheme in which each entity is associated with a pair of keys (public key and private key). In asymmetric cryptography, each time that a part of the keys is used to perform a cryptographic operation, the other part will be used for the opposite operation. For example, the public key is used in the encryption operation while the private key will be used for the decryption (opposite operation), the private key used for signature and public key for signature verification, etc. [Shirey, 2007].

Asymmetric algorithms are valuable over symmetric ones because:

- There is no need for exchanging keys, which eliminates the key distribution problem known in symmetric cryptography and eases the key management.
- It provides increased security since private keys do not ever need to be transmitted or revealed to anyone.
- It provides proof of non-repudiation.

The major drawback of public-key cryptography is that it consumes more computational time comparing with the symmetric one. Therefore, it is not always appropriate to use that kind of encryption with large amounts of data. However, an interesting approach would be to use public-key encryption to send a symmetric key, which it is going to be used in further data encryption operations.

#### 2.4.1.2 Access control

Access control is a security technique that aims to regulate who or what can view or use resources in a computing environment. There are two types of access control: physical and logical. Physical access control systems regulate the access to physical entities and resources such as access to campuses, buildings, rooms and physical IT assets. On the other hand, logical access control regulates the access to computer networks, system files and data. According to [Vincent C. Hu, 2014], both systems share a bunch of commonly used concepts that we are going to describe in what follows:

- **Object**: an entity that contains or receives information.
- **Subject**: an active entity (person, process, or device) that executes some tasks in the system.

- **Operation**: an active process invoked by a subject.
- **Permission (privilege)**: an authorization to perform some action on the system.

Since the main goal of this thesis is to ensure data security in cloud environment, we focus only on the logical access control systems that we present in what follows.

In order to protect the objects in a logical access control system, security administrators deploy access control mechanism that can be defined as *"The logical component that serves to receive the access request from the subject, to decide, and to enforce the access decision"* [Vincent C. Hu, 2014]. These mechanisms are usually founded based on access control models that are defined as follows [Vincent C. Hu, 2014]:

1. **Discretionary Access Control (DAC)**: is an access control model in which access rights can be defined according to the discretion of the object's owner or any entity that controls the object's access.
2. **Mandatory Access Control (MAC)**: is an access control model in which access rights are regulated by a central authority based on multiple levels of security.
3. **Identity-based access control (IBAC)**: is an access control model in which the system uses mechanisms such as access control lists (ACLs) to capture the identities of those allowed to access the object.
4. **Role-based access control (RBAC)**: is an access control model in which the system assigns a pre-defined role to each subject where each role carries a specific set of privileges.
5. **Attribute-based access control (ABAC)**: is an access control model in which the attributes are assigned to each user. ABAC systems evaluate access rights through a set of rules, policies and relationships using the attributes of users, systems and environmental conditions.

### 2.4.2 Cryptographic access control

Cryptographic access control is a paradigm designed for a global federation of information systems. This paradigm represents an access control mechanism that relies exclusively on cryptography to provide confidentiality and integrity of data managed in the system. Moreover, it allows to ensure reliable access control in untrusted environments, where the lack of global knowledge and control are defining

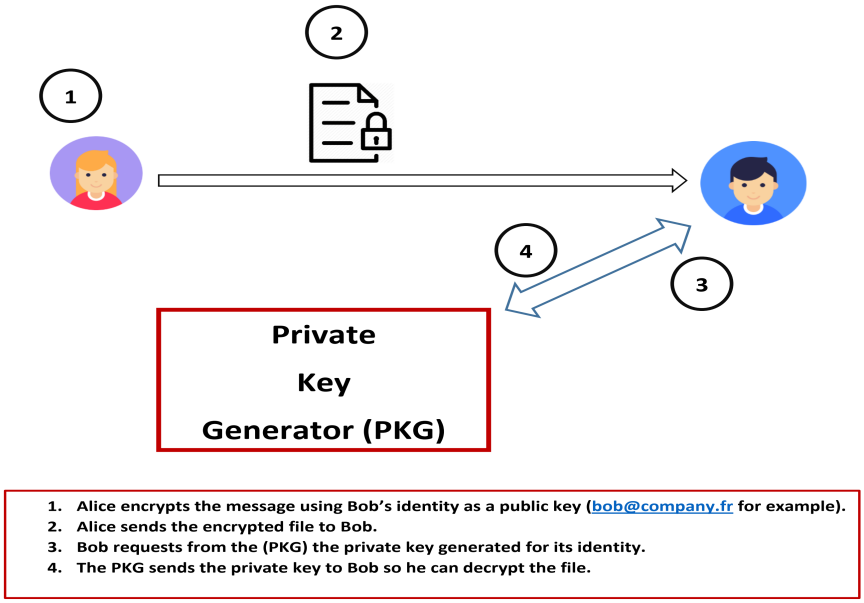



Figure 2.1 – Identity-based encryption

characteristics. In what follows, we present some of the most advanced cryptographic access control methods:

#### 2.4.2.1 Identity-based encryption (IBE)

IBE is an advanced public-key encryption method [Boneh and Franklin, 2001] in which the public key of a user is generated using identity information such as the user's email address. In IBE, a central trusted authority generates system parameters such as public/master pair of key, messages/ciphertexts spaces, etc., and publishes some of the generated system parameters (public parameters). A sender who has access to the public parameters of the system can encrypt a message using the receiver's unique information (email address for instance) as a key. On the other side, the receiver needs to obtain its decryption key from the central authority, that has established the public parameters, in order to successfully decrypt the received data. Figure 2.1, illustrates the functioning of IBE scheme.

#### 2.4.2.2 Fuzzy identity-based encryption (FIBE)

FIBE is a public-key encryption method that has been introduced as a new type of Identity-Based Encryption (IBE) scheme. In FIBE [Sahai and Waters, 2005], the user's identity is viewed as the set of descriptive attributes. FIBE scheme allows for a private key for an identity $\omega$, to decrypt a ciphertext encrypted with an identity $\omega'$, if and only if the identities $\omega$ and $\omega'$ are close to each other. The measure of closeness in this cryptographic scheme is based on set overlaps distance metric. For

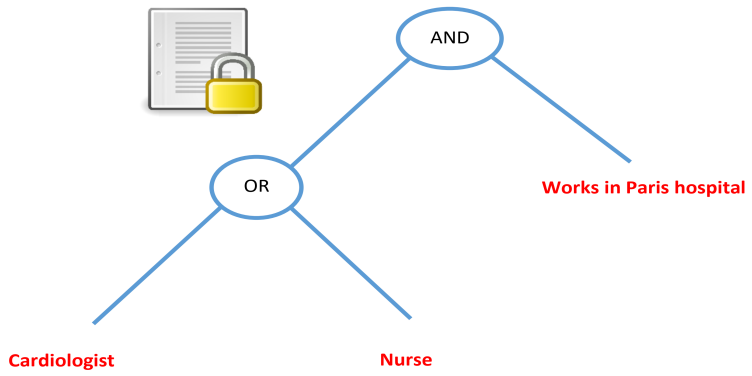



Figure 2.2 – attribute-based encryption access policy example

instance, in order to decrypt a message $C$ that has been encrypted with the public key $\omega'$, we need to have a private key for the identity $\omega$ with $|\omega \cap \omega'| \geq d$.

#### 2.4.2.3 Attribute-based encryption

Attribute-based encryption (ABE) is a relatively recent approach that reconsiders the concept of public-key cryptography. ABE goes one-step further compared to IBE, and defines the identity of the users as a set of attributes such as, e.g., roles, origins, abilities, etc. The main idea consists on encrypting the data using a set of attributes. In other words, the data owner encrypts its data based on an access policy that indicates the attributes that other users should possess in order to get access to the plaintext. Access policy (known also as access tree) is a logical expression combining several attributes through "$OR$", "$AND$", or other logical operators. Leaf nodes in the access tree represent attributes that the users should possess in their private keys, while non-leaf nodes represent threshold gates that are used to link between leaf nodes (attributes). Figure 2.2 shows an example of an access policy.

In Attribute-based encryption, we distinguish two main approaches, namely Key-policy Attribute-based encryption (KP-ABE) [Goyal et al., 2006] and Ciphertext-policy Attribute-based encryption (CP-ABE) [Bethencourt et al., 2007]. KP-ABE, associates ciphertexts with a set of descriptive attributes, while access policies are applied on the users' keys. Therefore, the encryptor exerts no control over who has access to its encrypted data, since he does not decide the access policy and its control is limited on the choice of the descriptive attributes associated with the ciphertext. On the other hand, CP-ABE uses access policies to encrypt data while users' secret keys are associated with a set of attributes. Thus, it offers more flexibility and most importantly, it gives more control to the data owner. Figure 2.3 illustrates the functioning of both KP-ABE and CP-ABE.

Since CP-ABE offers a more flexible access control, we will recite in what follows

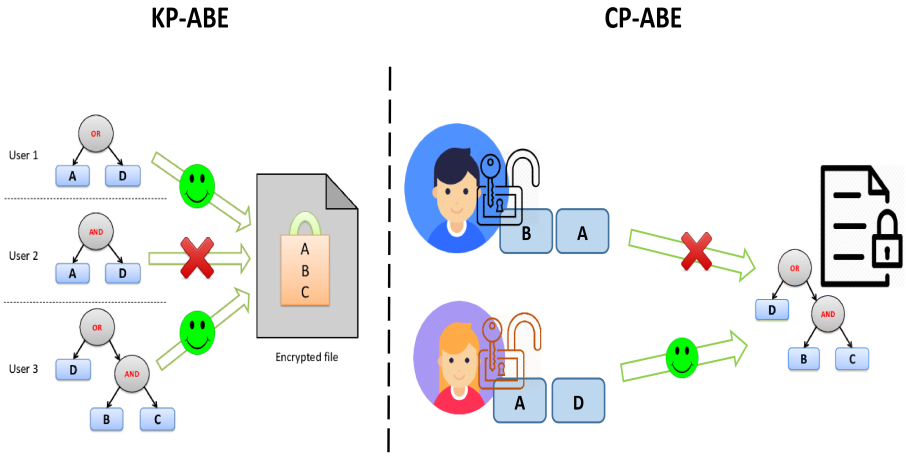



Figure 2.3 – KP-ABE vs CP-ABE

the construction steps of this cryptographic method [Bethencourt et al., 2007]:

**Setup:** in this phase, the algorithm chooses a bilinear group $G$ of prime order $p$ with a generator $g$. Next, it chooses two random values $\alpha$ and $\beta \in Z_p$ and generates the following public parameters:

$$G, g, h = g^{\beta}, e(g, g)^{\alpha}$$

And the master key:

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

**Key generation :** the algorithm uses the master key $MK$, a set of attributes $S$ to generate a private key for each user. The private key is computed using a random value $r \in Z_p^*$, which is unique to each user and a random value $r_j \in Z_p^*$ for each attribute $\lambda_j \in S$, the result is:

$$SK_t = (D = g^{(\alpha+r)/\beta}, \forall \lambda_j \in S : D_j = g^r.H(\lambda_j)^{r_j}, D'_j = g^{r_j})$$

**Message encryption :** when a user wants to upload data $M$ to the Cloud and shares it, he defines the tree access structure $T$ over the universe of attributes $L$, and encrypts the data under $T$.

Given an access structure $T$, for each node $x$ of $T$, the algorithm chooses a polynomial $q_x$ . These polynomials are chosen in a top-down mode, which is from the root node $R$. For each node in $T$, the degree dx of the polynomial $q_x$ will be set one less than the threshold value $k_x$ of that node, i.e. $d_x = k_x - 1$. Therefore, the root node $R$ is assigned with a random value $s \in Z_p^*$ and set $q_r(0) = s$, finally the root node $R$ sets $d_R$ and other random points to define $q_R$. Any other node $x$ sets:

$q_x(0) = q_{parent}(index(x))$ and randomly chooses $d_x$ and other points to define $q_x$. Next, the data owner uses the public parameters $PP$ and the tree of the access structure to encrypt the message $M$. Finally, the ciphertext will be:

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

**Delegation :** the delegation algorithm takes as parameters a secret key $SK$, which is specified for a set $S$ of attributes, and another set $\tilde{S}$ such that $\tilde{S} \subseteq S$. The algorithm chooses random $\tilde{r}$ and $\tilde{r}_k \forall k \in \tilde{S}$. Then it creates a new secret key as

$$\tilde{SK} = (\tilde{D} = Df^{\tilde{r}}, \forall k \in \tilde{S} : \tilde{D}_k = D_k g^{\tilde{r}}.H(k)^{\tilde{r_k}}, \tilde{D}'_k = D'_k g^{\tilde{r_k}})$$

**Message decryption:** data decryption in CP-ABE scheme is based on running up the recursive algorithm $DecryptNode(CT, SK, x)$, where $CT$ represents the ciphertext, $SK$ is the user's secret key which is associated with a set of attributes $S$ and $x$ is a node from the access tree $T$. If $x$ is a leaf node in $T$, let $i$ be the attribute contained on $x$, if $i \in S$ then:

$$\begin{aligned} DecryptNode(CT, SK, x) &= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ &= \frac{e(g^r.H(i)^{r_i}, h^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} \\ &= e(g, g)^{rq_x(0)} \end{aligned}$$

If $i \notin S$ then: DecryptNode $(CT, SK, x) = \perp$

In case where $x$ is not a leaf node. The algorithm $DecryptNode(CT, SK, x)$ will turn recursively from $x$ to his leaf children.

When a user wants to decrypt data, he simply calls the DecryptNode algorithm from the root $R$, if his set of attributes $S$ satisfies the access tree, he should get $e(g, g)^{\alpha s}$ by the end of the algorithm, then he must compute the following formula to get the plaintext:

$$\tilde{C}/(e(C, D)/A) = \tilde{C}/(e(h^s, g^{(\alpha+r)/\beta})/e(g, g)^{\alpha s}) = M$$

#### 2.4.2.4 Decentralized Attribute-based encryption:

In the first proposals of ABE scheme, it is assumed that a single authority manages all the users and attributes. Therefore, the users need to be within the same organization. In order to provide more flexibility and support attributes from

Tableau 2.1 – Comparison between existent Multi-authority schemes

| | **lin et al., 2008** | **Chase et al., 2009** | **Lewko et al., 2011** | **Chase, 2007** | **Muller et al., 2009** |
|---|---|---|---|---|---|
| Decentralization | + | + | + | − | − |
| No coordination between authorities | − | + | + | − | − |
| access policies expressiveness | + | − | + | − | + |
| Full collusion resistance | − | + | + | + | + |

different environment (several organizations), Chase [Chase, 2007] proposed a multi-authority attribute-based encryption scheme. However, this proposal relies on a central entity that ensures the coordination between several authorities. After that, several research work have been proposed for multi-authority architecture [Muller et al., 2009, Lewko and Waters, 2011] to achieve full distribution of authorities, while ensuring collusion resistance and the expressiveness of the access policies. Table 2.1, provides a comparison between all proposed decentralized ABE schemes.

As it is shown in table 2.1 Lewko's decentralized ABE [Lewko and Waters, 2011] is the most flexible solution among the other decentralized proposals. Thus, let us describe, in what follows, the different basis of this scheme:

**Global setup :** in this phase, the algorithm chooses a bilinear group $G$ of order $N = p_1p_2p_3$, then it publishes as global parameters the group $G$, a generator $g_1$ of $G_{p1}$ and a hash function used to map users' global identifier $GID$ to elements in $G$.

**Authority setup :** each authority chooses for each attribute $j$ two random values $\alpha_j$ and $\beta_j \in Z_N$ and generates the following public parameters:

$$Pk_i = \{e(g_1, g_1)^{\alpha_j}, g_1^{\beta_j}\}$$

and the master key:

$$MK = \{\beta_j, \alpha_j \ \forall j\}$$

**Key generation :** to create a key for attribute $j$ of a user identified by his $GID$, the authority responsible for that attribute computes the key as follows:

$$K_{j,GID} = g_1^{\alpha_j} H(GID)^{\beta_j}$$

**Message encryption :** the encryption algorithm takes as parameters a message

$M$, an access matrix $A$ with a function $\rho$ which maps its rows to attributes and a set of authorities' public keys. The first step of the encryption algorithm consists of choosing a random $s \in Z_N$ and two random vectors $\lambda$ and $\omega$: the first entry of $\lambda$ is set equal to $s$, while the first entry of $\omega$ is set to 0. Then, for each row $A_x$ in $A$, the algorithm sets $\omega_x = A_x \times \omega$ , and $\lambda_x = A_x \times \lambda$, and chooses a random $r_x$ in order to compute the Ciphertext as:

$$C_0 = Me(g_1, g_1)^s,$$
$$C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, g_1)^{\alpha_{\rho(x)} r_x},$$
$$C_{2,x} = g_1^{r_x}, C_{3,x} = g_1^{\beta_{\rho(x)} r_x} g_1^{\omega_x} \forall x$$

**Message decryption:** in order to decrypt the ciphertext, the decryptor needs to obtain his $GID$. Then, given an access matrix $A$ and a function $\rho$ which maps attributes to rows in $A$. According to the user's attribute keys $\{K_{\rho(x)}, GID\}$, the decryption algorithm selects the rows $A_x$ from $A$ such that $(1, 0, ..., 0)$ is in their span. After that the algorithm computes

$$\frac{C_{1,x} e(H(GID, C_{3,x}))}{e(K_{\rho(x)}, C_{2,x})} = e(g_1, g_1)^{\lambda(x)} e(H(GID), g_1)^{\omega(x)}$$

Next, the algorithm chooses constants $c_x \in Z_N$ such that

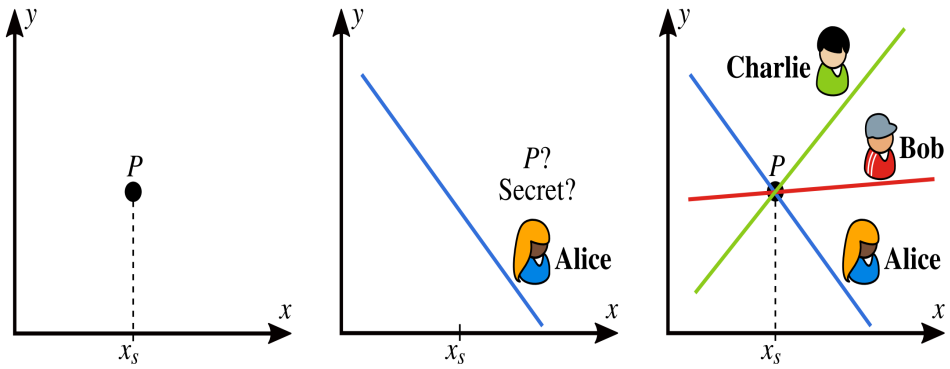
$$\sum_x c_x A_x = (1, 0, ..., 0)$$

Finally the obtained message $M$ is:

$$M = \frac{C_0}{\prod_x (e(g_1, g_1)^{\lambda(x)} e(H(GID), g_1)^{\omega(x)})^{c_x}} = \frac{C_0}{e(g_1, g_1)^s}$$

#### 2.4.2.5 Threshold cryptosystems:

In cryptography, a threshold-based system is a cryptosystem in which cryptographic functions such as encryption/decryption and signatures are distributed among a group of users. For instance, given $(t - n)$ threshold system, at least $t$ users need to combine their private keys in order to successfully decrypt the ciphertexts shared among the users. Usually, the major goal behind using threshold cryptography is to enhance a variety of security properties, such as confidentiality, access control, integrity and availability [Luis T. A. N. Brandao, 2018]. Secret sharing is a cornerstone technique in threshold cryptography. It enables a key (or some other




(a) Secret $x_s$ is P's $x$-coordinate (b) Single line does not reveal $x_s$ (c) Lines (shares) intersect at $P$

Figure 2.4 – Blakley's secret sharing [Luis T. A. N. Brandao, 2018]

secret input) to be split into multiple shares distributed across multiple parties. To reconstruct the key, the participant parties need to gather at least a threshold number of shares. In secret sharing, splitting a key into shares aims to protect the secrecy of the keys, since the leakage of one or few shares does not reveal the key. The secret sharing was invented independently by Blakley [Blakley, 1979] and Shamir [Shamir, 1979]. So, let us take a look on these two schemes:

1. **Blakley's secret sharing scheme**:

   Blakley's has based its secret sharing construction on the fact that any n nonparallel $(n-1)-$dimensional hyperplanes intersect at a specific point. For instance, two nonparallel lines in the same plane intersect at exactly one point. Therefore, the secret $S$ in Blakley's scheme is an x-coordinate $(x_S)$ of some predefined point $P(x_S, y_S)$. For threshold $k = 2$ in a system with $n$ users ($n = 3$ for example), given a secret $(x_S)$ of the point $P$ in the two-dimensional plane (see Fig.2.4(a)), a non-vertical line in the plane is defined as a set of points $(x, y)$ satisfying: $y = h\ timesx + g$ for some constants $h$ and $g$. If Alice obtains coefficients $h_A$ and $g_A$ for some line $y = h_A \times x + g_A$, containing the point $P$, this does not give her any advantage in discovering its $x_S$ (see Fig. 2.4(b)). Similarly, if Bob and Charlie obtain the coefficients of other lines that pass through the same point $P$, individually they cannot determine $P$. However, any two users (Alice-Bob for instance) combine their shares they can easily compute $P$ as the intersection of their lines (see Fig 2.4(c)).

2. **Shamir's secret sharing**:

   Shamir has based its secret sharing construction on the fact that that any set of $k$ distinct points determines a polynomial of degree $k-1$. Thus, in Shamir's

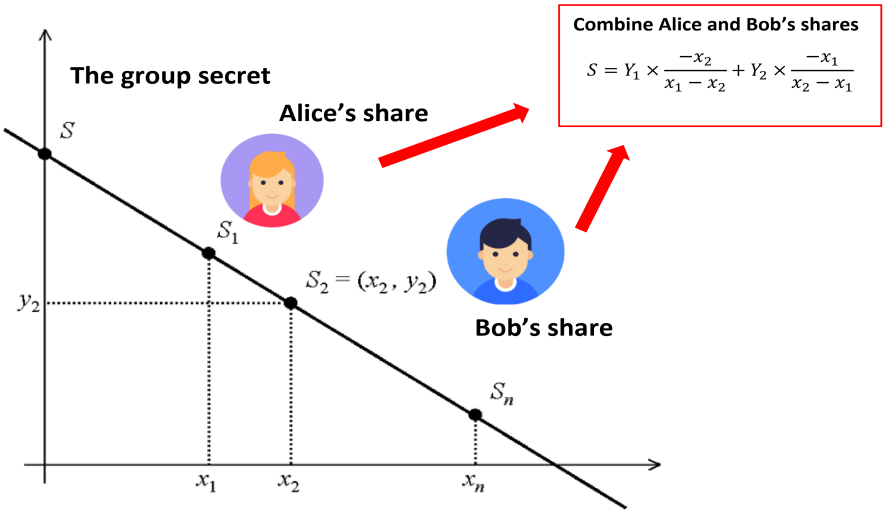



Figure 2.5 – Shamir's secret sharing

scheme, we first choose at random, $k-1$ positive integers $a_i, i \in [0, k-1]$, and build a polynomial $q$ as:

$$q(x) = a_0 + a_1 x + a_2 x^2 + ..... + a_{k-1} x^{k-1}$$

In which $a_0$ represents the secret $S$ and $(1, Sh_1 = q(1)), (2, Sh_2 = q(2)) \ldots (k, Sh_k = q(k))$ are the shares.

The polynomial $q$ can be reconstructed using Lagrange interpolation as:

$$q(x) = \sum_{i=1}^{k} Y_i \times \prod_{j=1, j \neq i}^{k} \frac{x - x_j}{x_i - x_j}$$

Where $Y_i = S_i$ Consequently, the secret $S$ can be calculated as $S = q(0)$

For instance, given a $(k = 2, n = 3)$ threshold system, if Alice obtains coefficients a point $P_A(x_A, y_A)$ generated through a polynomial $q$, this does not give Alice any advantage in discovering the polynomial $q$ neither the secret $q(0)$. Similarly, if Bob and Charlie obtain points generated through the same polynomial $q$, individually they cannot determine $q$. However, any two users (Alice-Bob for instance) combine their shares they can easily compute $q$ as Lagrange polynomial interpolation (see Figure 2.5).

## 2.5 Conclusion

In this chapter, we have introduced the main characteristics of cloud and fog computing paradigms. After that, we have presented a brief definition of security and its main services. Next, we have focused on cryptography and access control which are the main security mechanisms used in the work being done during this thesis. First, we discussed the existing classifications of both cryptographic methods and access control models. Later on, we have provided a description to several advanced cryptographic solutions, proposed in the literature, and which have been used to ensure access control in data sharing applications.

# *Cloud/Fog computing security: issues and challenges*

## 3.1 Introduction

Cloud computing has proven its efficiency by offering a flexible and ubiquitous data management, storage and computing services. However, in spite of the advantages that this technology offers, it also introduces several challenging issues. In January 2018, RightScale conducted its annual State of the Cloud Survey [rig, ] on the latest cloud trends. They questioned 997 technical professionals across a broad cross-section of organizations about their adoption of cloud infrastructure. Their findings were insightful, especially in regards to current cloud computing challenges. According to the report results, security has been pointed as the top challenge in cloud computing. Indeed, security is a crosscutting function that spans all layers of cloud architectures, involving several security levels going from physical to application security.

To ensure security in cloud environments, both cloud providers and consumers need to address several issues in relation with security requirements such as authentication, authorization, availability, confidentiality, identity management, integrity, audit, security monitoring, incident response, and security policy management [Hogan and Sokol, ].

In this chapter, we first give an overview on cloud computing security issues in section 3.2. Then, we summarize the most important challenges facing cloud providers and consumers in section 3.3. In section 3.4, we focus on the cryptographic access control challenge, we introduce the revocation problem statement and its related works. Later on, we present in section 3.5 some additional challenges in relation with fog computing security. Then, we introduce the mutual authentication and the traceable privacy-preserving problems along with their related works in sections 3.6 and 3.7 respectively. Finally, we conclude in section 3.8.

## 3.2 An overview on Cloud computing security issues

In this section, we present the most important cloud security issues that have been mentioned in the literature. We classify these issues according to different contexts as follows:

### 3.2.1 Data related issues

In the cloud computing model, users need to externalize their data into cloud provider's data centers. Therefore, they will not have any control over their stored data because they will not be physically located at the same location as their data. Since cloud computing provides a pool of servers to store users' data, storage service is usually ensured on the basis of multi-location feature, where data is stored in different servers across the world. Consequently, this approach can bring new security threats and legal problems, since different storage locations also refer to different data protection policies. Furthermore, it is hard to check data integrity and confidentiality since these tasks are time consuming, especially if the amount of externalized data is huge (Big data environment). In what follows, we discuss the most known issues related to data security.

1. **Data availability issues**: the physical and virtual resources (database and processing servers) of the cloud are supposed to ensure a high availability features. To achieve availability and scalability of data and services, cloud providers' systems usually run applications on multiple servers. This approach is fault tolerant. Indeed, if an application crashes in some server then another same application server is present to ensure data and service continuation and availability. However, this approach also might enable DOS attacks, since it is also possible that a server has highly demanding application tasks and due to duplication, it will definitely consume more power, occupy more resources, and take more time in the processing. Availability issues could also occur due to hardware failure. Indeed, a single fault can lead to a partial or a complete failure of the system, and directly affects the availability of data and services [Singh et al., 2016] [Barona and Anita, 2017].

2. **Data recovery issues**: cloud computing is a resource pooling technology with elasticity feature that allows to allocate dynamic and on demand resources according to users' requests. Therefore, the resources that are allocated to a user at some moment $t$ may possibly be allocated to another user at

some later point of time. Despite the benefits of elasticity in the cloud's resource management model, some questions arise about possible data recovery operation that definitely violates data confidentiality. Indeed, a malicious user can try to use data recovery mechanisms in order to recover the previous user data, and uses some sensitive data for malicious purposes [Barona and Anita, 2017] [Modi et al., 2013].

3. **Data Breaches**: a cloud computing is a multi-tenant environment in which computing resources are shared between various users and organizations. Therefore, customers using different applications on virtual machines could share the same database and consequently, any corruption event that occurs in this shared data space will definitely affect all the users sharing the same space [Los et al., 2013]. Therefore, an attacker may exploit the multi-tenancy feature and try to breach other users' data located in the same physical machine in case where this data is not properly isolated. Moreover, data leakage may also be the result of other external threats such as malicious hacks of cloud providers, malwares and compromises of cloud user accounts. In addition to external threats, data leakage can also be due to malicious insider behaviors. Indeed, cloud is a third-party service model, which means that data is potentially at risk of being viewed or mishandled by providers or some of their employees.

### 3.2.2 Virtualization related issues

Cloud computing has been known by multi-tenancy and virtualization features that provide more profit for both the users and the cloud providers. However, these features expose cloud computing technology to several threats. Indeed, to better manage computing resources in the cloud model, the virtualization, system often changes the state of the virtual machines and store them in a database repository. Suspended VM's (dormant) could represent a potential security threat for virtualization system. In fact, when a virtual machine goes offline, the security software updates and the deployment of critical code patches stop happening. Therefore, it makes them out of date and thus become a temporary point of vulnerability during the period between their brought online and the next patches and software updates.

VM sprawl is another security issue related to the virtualization. This issue refers to the situation when the number of virtual machines (VMs) on the cloud reaches a point where the administrator can no longer manage them effectively. Indeed, a bad management of the virtualized environment will definitely cause crashes in the cloud due to system low resources. In addition, it may also raise the risk of rogue

virtual machines that try to create havoc in the IT infrastructure.

Moreover, an attacker may perform co-location attack to access other VM's data in cases where there is no proper logical and virtual isolation between VMs. In fact, when the attacker and the target virtual machines are located on the same host and under the control of the same hypervisor, several attacks might occur. These attacks mainly aim to extract information from the target VM by noticing patterns of resource usage, particularly CPU usage. For instance, an attacker may check if the access to the target increases the rate of cache misses in the attacker's VM; if it does, then he concludes that they are sharing the same hardware. Therefore, he can exploit timing and cache interference effects between VMs to extract information from the target's VM [Booth et al., 2013].

In addition, the Virtual Machine Monitor VMM (known as the hypervisor) might also represent a single point of failure to the whole virtualization system. Although there are no known attacks that have been yet reported in the hypervisors, the threat is still very real. A hypervisor is the virtualization management software that controls all the virtual machines on a single physical server. Normally, each guest machine has its proper virtual space and it is not expressly allowed to access neither to the space of another virtual machine, nor to the space of the hypervisor itself. However, many experiments have proven that the hypervisor can be subject of some threats such as hyperjacking, escaping a virtual machine, and web-based hypervisor consoles. Hypervisor's management console can also be subject to several attacks such as Cross-site scripting and SQL injections, in addition to rootkits (BluePill for example [Singh et al., 2016]) that might compromise the hypervisor and gives the attacker full control of the physical machine.

### 3.2.3 Identity and access management issues

Managing users' identities and access rights to applications and data in cloud environment is increasingly important, especially as the number and complexity of laws and regulations grow. Indeed, control of access rights plays a unique role in cloud computing, because the data is no longer stored on devices managed by the data owners. Thus, cloud providers must provide efficient systems that handle identities and access management for a growing number of users, both inside and outside the organization, without compromising security or exposing sensitive information. Identity and Access Management systems (IAM) aim to provide the right people with the right access at the right time. As more and more organizations are adding cloud services to their infrastructure, the process involved in the management of identities is getting complicated. In fact, these systems are facing several issues

in relation with identity theft and access rights violation. These issues may cause serious damage to sensitive data externalized into the cloud as it has been illustrated in what follows:

1. **User credentials issues**: in cloud computing, the providers usually use access directories such as Lightweight Directory Access Protocol (LDAP) or Microsoft Active Directory (AD) technology to manage users' credentials. Therefore, the security of the directory becomes an important task since it represents a critical point in the system. Weak password recovery mechanism and credentials reset represent another threat for these systems. Indeed, if an attacker succeeds to steal users' credentials, then he can access all the credits and sensitive information and manipulate whatever the target user was allowed to access [Singh et al., 2016].

2. **Identity management and authentication issues**: the Identity Management (IDM) is a mechanism to identify and manage users' identities, cloud objects, organization's accounts and provide access to the resources according to the administrative policies. Beside traditional IDM approaches, new ones such as credential synchronization and federated identification are available in cloud computing model nowadays. Still, existing cloud IDM systems have some issues to solve. One can cite account information leakage threat, trust, validation and interoperability issues that happen from using different identity tokens and protocols. On the other hand, the authentication represent the first step to achieve a secure access to cloud applications. Therefore, weak authentication mechanisms make cloud applications subject to several threats such as brute-force and dictionary attacks. There are several authentication techniques used in cloud systems such as simple text based passwords, one time password, graphical password, third party authentication, 3D object password, and biometric password. However, most of these authentication techniques might not provide sufficient security level due to users' behaviors in the first place. Indeed, users usually use the same credentials to get access on any system (same password, for example) and thus, if an attacker discovers a weak point in any provider's system and get access to the victim's credential, then he will be able to impersonate that victim and get access to its data in other cloud applications.[Singh and Chatterjee, 2017]

3. **Authorization management issues**: the authorization is the process of granting or denying permissions to a person or a service in the system. Access control management standards as XACML are widely used in the cloud computing environment since it allows to state policies and form access

control decisions. However, these standards do not define protocols, transport mechanisms, or specify how users' credentials are validated. In addition, messages transmitted between XACML entities are susceptible to unauthorized disclosure, replay, deletion and modification attacks, unless sufficient safeguards are in place to protect transactions [Jansen and Grance, 2011]. Moreover, in cloud computing model, the data owner loses the control of its data once it is uploaded into the cloud. Therefore, he will not be able to actually define and survey the access authorizations above its data. Indeed, the access to information must be strictly limited to authorized users in order to guarantee data confidentiality. However, since access policies are applied by the cloud, even if it manages somehow to ensure sufficient protection of users' data against data leakage threats, it will be hard to ensure its protection against malicious insiders [Singh et al., 2016].

### 3.2.4 Malicious insider issue

Cloud services have vastly expanded the scope of malicious insider threat. Generally, a malicious insider is a person who has the appropriate access rights to an information system and misuses his privileges [Bishop and Gates, 2008]. Regardless how reliable are the technical and operational counter measurements deployed to defend against external malicious actions, it remains useless if it does not consider the potential threat that might come from entities within the computing system. The insider threat affects every infrastructure and cause significantly more damage to the organization than any external threat. Indeed, as the attack can affect a large number of cloud users, the impact of such attack will be significant. For instance, an administrator responsible for performing regular backups of the systems where client resources are hosted (virtual machines, data stores), could exploit the fact that he has access to backups and thus exploit sensitive users' data. In the case of cloud computing, the malicious insider can also be an attacker who works for an organization that is using cloud services and who lead attacks against the cloud infrastructure or its own organization. In both situations, detecting such indirect attacks, is a challenging task [Kandias et al., ].

### 3.2.5 Software related issues

Vulnerabilities in cloud computing software represent another concern for both customers and providers. Indeed, developers write each software program from a personal way of thinking, using different programming language, without any common coding rules or guidance in term of isolation between platforms, safe thread

termination, resource monitoring, uncertain system calls, etc. Therefore, an attacker may get access to the software source code and exploit a vulnerability in that code to get access to the user's data or take control over the virtual machine. Besides that, the incapability of the software to tolerate faults can also be an issue that might cause availability problems. For instance, the recent amazon S3 incident [Ama, ] showed that their software was not able to tolerate mistakes in administrators' commands and due to that, all the system restarted and several services went offline. The providers cannot easily detect fault tolerance vulnerabilities and thus if an attacker suddenly comes to find out such a vulnerability, the outcome might be tragic.

### 3.2.6 Internet and web technology issues

The cloud computing is an internet-based system, which makes it subject to every kind of threat known in the internet environment. Indeed, cloud computing services are usually accessed and managed using web standards and thus it may expose both the users and the providers to several security issues related to these standards. One can cite HTTP session riding and session hijacking, Man in the middle (MITM) attacks, IP spoofing, port scanning, malware injections, and packet sniffing. Moreover, web services can also be subject to several security threats, especially if the providers deliver their services through insecure web APIs. In [McIntosh and Austel, 2005], the authors described an XML-based signature wrapping and rewriting attack that target SOAP messages, and allows to access the web resources through the injection of forged messages in the XML field. In 2009, Researchers from MIT and UC San Diego demonstrated an attack against Amazon's EC2 in which the eavesdropper can have access to multiple Amazon EC2 services [Talbot, 2009].

Web technologies used by the cloud to provide computing services are also facing a considerable amount of security issues. In fact, malicious web links and web sites continue to spread malware in the web environment through several malicious attacks such as the Cross site scripting (XSS), code injection, broken authentication, session management, etc.[Wichers, ]. Cookies theft and HTML hidden field attacks are other threats that may expose users' privacy while they are using cloud computing services. Indeed, users may browse social network sites, personal email accounts, and other application site at the same period of time they are using cloud services. This extra browsing may be the entering point for malware that might steal cookies from the users' browser and breaches its privacy.

### 3.2.7 Privacy issues

In general, users' privacy protection refers to the protection of users' personal information. Nist [Erika McCallister, 2010] defines the personal information as "*any information about an individual maintained by an agency, including (1) any information that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records; and (2) any other information that is linked or linkable to an individual, such as medical, educational, financial, and employment information*".

Despite the advantages that the users gain with cloud computing model, major concerns raise in the prospect. One of these concerns is the privacy of outsourced data in cloud [Jaeger and Schiffman, 2010]. Indeed, sensitive information such as e-mail, health records, etc., may fall in the hand of unauthorized users or even be hacked [Brunette et al., 2009]. Due to the open nature of cloud platforms, users' personal information can be subject of attacks, not only from unauthorized outsiders but also from malicious insiders. Privacy violation risk is not limited on getting access to sensitive information; it can also be extended to users' activity tracking, data analysis and record linkage, which can be used for advertising purposes without users' authorization.

## 3.3 Challenges

From the security issues discussed above, we can summarize the security challenges facing the researchers as follows:

- **Access control**: access control is one of the most important security challenges that both cloud providers and users should ensure in their data sharing systems. Access control challenge covers several security issues such as data breaches, data recovery issues, identity and authorization management, credentials assignment and users revocation.

- **Data availability**: data availability refers to the ability of data to remain accessible all the time. This challenge covers security issues related to the physical reliability of the infrastructure and the ability to recover corrupted data.

- **Management of virtualization environment**: the management of virtualized environment is a cornerstone challenge for cloud providers. Indeed, the providers should ensure full VMs isolation and deal efficiently with VM's scaling issues.

- **Cloud interoperability**: interoperability and the ability to share various types of information between clouds is a challenge that concerns several fields including security. This broad area of cloud interoperability is usually known as cloud federation. In terms of security, the cloud federation paradigm introduces new challenges in relation with the verification of users' authenticity, and the management of access control across cloud providers.

- **Network security**: since cloud computing is a technology that requires network access, preventing network related attacks such as IP spoofing, MITM, Cross-Site Scripting, Phishing, etc., remains a challenge that need to be addressed by both users and cloud providers.

- **Privacy**: privacy is another important challenge in the cloud computing environment. Indeed, data externalization in the cloud servers might expose users' personal information to leakage threat, or might be subject to detection of users' behavior pattern, activity tracking, interests and preference detection.

In addition to security challenges, the deployment of security appliances in data centers should not affect the performance of the cloud applications. On the other hand, security features proposed by the cloud providers should satisfy the predefined service level agreements (SLAs). Moreover, it should be flexible in a manner that allows cloud applications to exploit their security functionalities. Therefore, setting up flexible and reliable security solutions in both users and cloud provider sides is also a challenge in the cloud computing environment.

## 3.4 Problem 1: Revocation management in attribute-based access control environement

Access control is one of the most important challenges to tackle in cloud computing environments. This challenge consists on attributing access rights to the users, managing revocation situations, and finally, ensuring the effectiveness of the whole access control process.

As seen in chapter 2, several access control models can be applied in data-sharing context. Most of these models consist of relying on service providers (Cloud providers [ama, 2018, azu, , dro, ] for example) to manage users' identities, attribute access rights and then supervise the access to the data [Singh and Chatterjee, 2017]; or using authentication servers such as Kerberos [Neuman and Ts'o, 1994], Radius [Rigney et al., 2000], etc. However, relying on service providers to ensure access control requires from the users to trust those servers. In fact, when the service

provider manages the whole access control process by itself, the data owner will not have effective control over its data, while the provider will have full control. Hence, an access control solution that protects the data from service providers' misbehaviour and gives the data owner more control, power on its data, will be more suitable for data-sharing in the cloud computing environement.

Ciphertext-policy Attribute-based encryption (CP-ABE) [Bethencourt et al., 2007] is a promising cryptographic method which provides flexible and fine-grained access control. CP-ABE ensures a high confidentiality level and allows to define access policies based on users' roles. Indeed, in this method, the data owner defines which set of attributes a user must have in order to successfully decrypt the ciphertext. Therefore, it gives him some control on his own data and avoids him to rely on any service provider in the access control process.

In the first proposals of ABE scheme, it is assumed that all the users and attributes are managed by a single authority. So, the users need to be within the same organization. In order to provide more flexibility and support attributes from different environment (several organizations), Chase [Chase, 2007] proposed a multi-authority attribute based encryption scheme. However, this proposal relies on a central entity which ensures the coordination between several authorities. After that, several research work have been proposed for multi-authority architecture [Muller et al., 2009, Lewko and Waters, 2011] to achieve full distribution of authorities, while ensuring collusion resistance and the expressiveness of the access policies.

The adoption of CP-ABE either in centralized or decentralized models introduces several challenges. One of the major challenges of this method is users and attributes revocation. Indeed, since users in attribute-based systems possess attributes in common, the revocation of an attribute from a user's key affects all the users who possess keys with the same revoked attribute. So, the challenging problem can be stated as follows: given an environment where each entity is characterized by a set of attributes issued by different authorities and uses ABE as an access control method, how can the authorities banish some attributes from an entity's key while ensuring a minimum computational cost?

### 3.4.1 Related work

As mentioned before, the main problem of attribute-based encryption is the revocation. Through the last few years, the revocation problem has been addressed in several research papers. Most of these papers treated the revocation in the central model of ABE, but few focussed on decentralized ABE. We discuss in what follows, the revocation solutions proposed in the literature.

#### 3.4.1.1 Centralized revocation solutions

Several centralized revocation solutions have been proposed in the literature. Some proposals provide only a time-based revocation while others could perform an immediate revocation.

In time-based solution [Pirretti et al., 2010] , the system assigns an expiration date to each attribute. Bethencourt et al. [Bethencourt et al., 2007] proposed to define one expiration date for the secret key instead of each attribute. In both systems, a generation of new keys is launched as soon as the expiration time is overtaken. In addition, the revocation cannot be applied till the next expiry date. Thus, during this period of time, revoked users may continue to successfully decrypt the data because the new keys are not yet distributed. Moreover, the re-keying process can be very expensive in terms of computational time especially in large scale systems.

Ibrahimi et al. [Ibraimi et al., 2009] and Yu et al. [Yu et al., 2010] proposed an immediate revocation scheme which includes a semi-trusted third party (proxy) in the architecture. However, their solutions do not achieve the fine-grained access because the users do not rely only on their attributes to get access, they also need a partial decryption from the proxy as well.

Borgh et al. [Borgh et al., 2017] also proposed a proxy based revocation for constrained devices. Ostrovsky et al. [Ostrovsky et al., 2007], proposed a revocation scheme in which the data owner adds the negation of revoked users' identities to the access policy which is not very efficient. Indeed, encryption and decryption overheads grow up with the number of revocked users. In addition, the data owners should possess the revocation list, since revoked users' identities are included in the access policy.

Golle et al. [Staddon et al., 2008] proposed a revocation scheme based on KP-ABE and which works only with a specific number of attributes associated to the ciphertext.

Attrapadung et al. [Attrapadung and Imai, 2009] proposed a direct revocation scheme where they combined broadcast encryption and ABE. However, this approach forces the data owner to maintain the membership list, which is not applicable in all cases.

Junod's et al. [Junod and Karlov, 2010], proposed to include the identities of non-revoked users in the access policy, then they update these identities attributes to achieve revocation. However, each time that a new user joins the system, it is necessary to include its identity in the ciphertext and so it grows up with new users incoming.

Lewko et al. [Lewko et al., 2010] proposed a revocation scheme in a system with small size public and private keys. However, their approach also increases the size of the ciphertext because they incorporate the list of revoked and non-revoked users into it.

Xu et al. [Xu and Martin, 2012] proposed a dynamic revocation scheme which is similar to the proxy based solutions. In this proposal, the Cloud server maintains the revocation list and performs a re-encryption of the uploaded data using a delegation key. When a user requests the data, the Cloud server decrypts the re-encrypted data only if that user is not in the revocation list. This solution gives the cloud server a full control on the revocation list and once the user is revoked he loses all his access rights.

Hur and Noh [Hur and Noh, 2011] introduced the key encryption keys (KEK) idea to realize revocation. In their scheme, the storage server generates KEKs by setting up a binary tree in which the leaf nodes represent the users, and each user receives the path key from his leaf to the root. Then, the ciphertext is re-encrypted using the attributes group KEK's, and when a user is revoked an update on the KEK's will be launched. However, the problem with this solution is in the management of the binary tree, which becomes hard when the number of joining or leaving users raises.

In [Yan and Shi, 2017], the authors proposed a cooperation between the data owner and the Cloud service provider to perform the revocation. They suggest that the data owner generates new keys for the users which might cost him a huge computational time.

In [Cui et al., 2018], the authors proposed a revocation scheme in which the users need to perform a new registration each time that the revocation occurs.

Liu et al. [Liu et al., 2017] defined a version for each attribute, and proposed to change this version and perform a key update to achieve attribute revocation.

#### 3.4.1.2 Decentralized revocation solutions

Yang et al. [Yang et al., 2012] proposed a temporal attribute-based access control on a multi-authority model in which the validity of the attributes depends on time slots. Thus, only the users who possess the attributes on the current time slot could access the data. In this solution, the data is divided into several granularities and each part is encrypted according to an access policy. The users' keys are composed of two part of keys, called the secret part and the update part. The secret part ties the attributes with the users' global identifier and does not allow the decryption by itself. On the other hand, the update part ties the time slot to the attributes. The

combination of both part of keys allows the user to succesfully decrypt the data. The revocation is performed at the end of the time slot by updating non-revoked users' the part of the key which is linked to the time slot (the update part).

Similarly to [Bethencourt et al., 2007, Pirretti et al., 2010], if a user is revoked before the expiration of the time slot, he will be able to decrypt the data until the end of the time slot, which causes a security degradation.

De et al. [De and Ruj, 2017] realized the revocation on decentralized ABE by relying on a semi-trusted proxy, which possess a partial decryption key and a revocation list. In this architecture, the decryption process includes the proxy and the data requester. The proxy realizes the revocation by adding revoked users to its revocation list. Thus, before performing any partial decryption, the proxy checks if the data requester is not in the revocation list. This solution realizes immediate revocation on both attributes and users levels, but as the centralized proxy solutions [Ibraimi et al., 2009, Yu et al., 2010], it does not achieve the fine-grained access.

Huang et al. [Huang et al., 2015] suggest to manage the revocation by deleting the revoked attribute from the access policy, which is not practical, since this solution causes the revocation of all the users who have the revoked attribute.

Yang et al. and Hong et al. [Yang and Jia, 2014, Zhong et al., 2018] used attributes version to realize a multi-authority access control with an efficient revocation. In their solution, they assign a version to each attribute, and when an attribute is revoked, the authority updates the version of that attribute and launches an update on the non-revoked users' keys and re-encrypts the ciphertext as well. However, the problem of this solution is the computational cost of keys update and ciphertext re-encryption.

Ruj et al. [Ruj et al., 2011] proposed "DACC", a distributed access control scheme which uses Lewko's et al. [Lewko and Waters, 2011] decentralized CP-ABE. This solution achieves the revocation by giving a part of the ciphertext to the non-revoked users. However, it requires a communication between the owner and the users each time that the revocation occurs. Besides, the non-revoked users might store multiple parts of all the ciphertexts that they are allowed to access, which is not convenient in the case where the storage capacity of users' devices is limited.

## 3.5 Fog computing security challenges

Fog computing is a new paradigm, which extends cloud computing services to the edge of the network. This new architecture integrates network edge devices to overcome several cloud computing limitations related to bandwidth and latency.

In terms of security, fog computing inherits several challenges from cloud

technology. These challenges are mainly related to data protection, access control, virtualization management, network security, etc. Moreover, new challenges that used to easily be managed in cloud computing become much harder in fog technology. Among these challenges, one can cite:

- **Authentication**: in fog computing architecture, we move on from a simple authentication scheme, where only the service provider needs to verify the identity of the users, to a mutual authentication scheme. In fact, the untrustworthy nature of fog architecture makes it susceptible to several attacks as Man in the middle (MITM), that aim to impersonate legitimate fog nodes or users in order to get confidential information. Therefore, setting up a robust authentication mechanism that allows both the users and fog nodes to verify the authenticity of each other and deal efficiently with attacks, such as MITM, is one of the most important challenges to address. In addition, since fog computing paradigm aims to overcome cloud computing shortcomings in terms of latency and bandwidth saving, proposing authentication solutions that do not rely too much on the cloud or any central authentication servers is another part of the authentication challenge in this new architecture [Hu et al., 2017].

- **Trust management**: fog computing extends cloud services by pooling the local resources of the network, which adds a resource-rich extra layer composed of a large number of edge devices that provides services at the edge of the network. However, security protocols in these fog nodes are without doubts less robust than protocols set up in the cloud. Therefore, fog nodes are susceptive to act maliciously. Thus, the presence of a system that manages trustworthiness will allow the users to get a global view on dishonest nodes in the network. There are many challenges, though. Users should be able to exchange compatible trust information with each other all over the architecture, even if it is located in different trust domains. The storage and dissemination of trust information, is another problem that needs to be solved. Indeed, due to the massive size of fog computing architecture, there will be a huge amount of trust information generated by the users. Thus, it is a challenge to manage this information, to store it and to make it accessible anywhere, anytime, with as less latency as possible. Moreover, dealing with attacks that aim to manipulate trust values such as badmouthing and self promoting attacks, collaborative attacks, etc., is also a crucial requirement in any trust management protocol [Roman et al., 2018].

- **Intrusion detection systems (IDS)**: usually an intrusion detection system is implemented in architectures which has homogenous components so the only

challenge was to understand the possible attacks against the components and then to implement procedures able to detect the signature of these attacks. However, in fog computing architecture, components are heterogenous and thus, the attacks that might target them are completely different from a component to another. Therefore, the implemented intrusion detection system needs to adapt according to the component nature. Thus, we will most likely have several intrusion detection systems cooperating together rather than a global one. The main challenges in this case consist of understanding the new attacks that might be launched in fog computing architecture; ensuring the interoperability between the different IDSs; having a global monitoring infrastructure that allows to detect attacks in a large scale, based on information delivered by the multiple IDSs set up at the edge of the network; and finally making these systems as autonomous as possible [Roman et al., 2018].

- **Distributed denial of service (DDOS)**: since fog servers are resource constraint, it will be very difficult to deal with large number of irrelevant requests simultaneously. As a result, resources for hosting legitimate services become unavailable and cause service interruptions. Besides, fog servers can also be used to launch DDOS attacks since it can be easy to compromise them. Similar attacks have been witnessed recently [Cyb, ], where a group of hackers used internet-connected home devices to launch DDOS attacks against popular websites such as PayPal, Twitter, Spotify, etc., and have more computational capabilities in fog servers will definitely rise the possibilities to perform the same cooperative attacks in fog computing. Therefore, DDOS is a challenge that needs to be well addressed in any future fog computing standardization.

## 3.6 Problem 2: Mutual authentication in fog computing architecture

With the new fog-computing paradigm, new challenges appear in prospect. Data security is one of the most important challenges of this architecture. Indeed, the fully distributed and untrustworthy nature of this architecture makes data security as one of the main users' concerns [Kumar, 2010a].

Authentication service is the entry point of any security system, and which consists of verifying users' identities. Authentication protocols can be ranged in three main families [Brainard et al., 2006]:

- **"Something you know"** protocols: such as passwords-based authentication.
- **"Something you have"** protocols: such as certificate-based authentication.
- **"Something you are"** protocols: such as biometric authentication.

Using authentication based on passwords in fog computing architecture has some serious shortcomings. Indeed, these systems are not robust and do not provide a high security level. Moreover, it has not been adapted to achieve mutual authentication since even if the fog nodes have a storage capacity that allows them to store users logins and passwords, it is not common that a user authenticates fog nodes using the same mechanism. Similarly to password based authentication, solutions that use biometric information cannot not be adopted in fog computing paradigm as well. In fact, the fog computing architecture is known with its heterogeneous components, which do not have biometric definition such as, connected objects, autonomous cars, fog nodes, etc.

As we can notice, "Something you have" authentication schemes are the most convenient to this architecture. Certificate-based authentication is an efficient authentication scheme which is used in several applications to verify the identity of any system component.

When an end entity uses a certificate, a trust relationship must be verified between the end entity certificate and the root certificate authority. This trust relationship is verified by validating the contents of all of the certificates in the certificate chain up to the root certificate authority. Indeed, in most situations, an entity gets its certificate from an intermediate certificate authority which got its certificate from one of the trusted root authorities or from a chain of intermediate authorities ascending to one of the trusted root authorities. However, to verify the validity of a given certificate, it is necessary to verify a set of certificates across a multi-level layer going up to one of the trusted root certificates.

In an heterogeneous fog computing architecture, most users' and fog nodes certificates come from different authorities. Thus, to mutually authenticate each other, both users and fog nodes will find themselves claiming certificates from intermediate authorities, which are in the path chain ascending to one of the trusted certificate authorities regarding the user or the fog node. Claiming certificates to ensure mutual authentication, each time that a user requests a service from a fog node will definitely cause latency issues. This is contradictory with fog computing paradigm, which aims in the first place to ensure low latency levels by extending cloud services into the edge of the network. Moreover, certificate-based solutions suffer from scalability issues since the central authority might need to handle a

huge number of verification requests. Revocation management is another issue in certificate-based solutions. Indeed, the users have to frequently download and store the most recent certificate revocation list (CRL) from each relevant authority in order to verify the validity of signatures of other entities [Alrawais et al., 2017]. Additionally, the cumulative number of revoked certificates makes the CRL file size grows over time, which will endure a significant communication and storage overhead at the user side. Finally, the certificate authority constitutes a single point of failure, which makes it subject of several cyber attacks (Diginotar incident [Leavitt, 2011]).

### 3.6.1 Related work

Generally, authentication is the first service which needs to be addressed in any security system. As far as we know there have been only one scientific paper [Ibrahim, 2016] which addresses mutual authentication in fog computing. In [Ibrahim, 2016], the author proposed an authentication scheme that allows any Fog user to authenticate mutually with any Fog server under the authority of a Cloud service provider. In this scheme, a Registration Authority (RA) is set up in the cloud and defines a random master key for each user. This master key is used to generate secret keys for each fog server in order to allow them to verify the authenticity of the users. Thus, each fog server will maintain a secret key for each user in the network. Moreover, each time a user joins the network, the RA generates and sends a secret key to each fog server. Otherwise, the fog servers are not going to be able to authenticate that new user and thus the user will not be able to access the fog server services. In addition, the author in [Ibrahim, 2016] did not consider authentication between fog servers.

There have been several solutions proposed for similar architecture as fog computing. In [Balfanz et al., 2002] the authors proposed an authentication scheme based on near field communication (NFC) technologies, which relies on physical contact for pre-authentication in a location-limited channel. Similarly, NFC-based solutions have been used as an authentication model for Cloudlet in [Bouzefrane et al., 2014]. However, this solution cannot always be applied, since there is no guarantee that the users and the fog nodes are located in a near area. Similarly, password based solutions have been proposed in several architectures [Kumar, 2010b] [Lu et al., 2008] [Panayappan et al., 2007]. The problem with these solutions is their low entropy. Indeed, these solutions are vulnerable to dictionary attacks. Moreover, due to the untrustworthy nature of fog architecture, the fog nodes cannot be trusted with users login and passwords. In addition, solutions based on passwords cannot ensure mutual authentication by themselves. Likewise,

Biometric authentication techniques are complex and cannot always be applied in fog computing due to the heterogeneous nature of fog architecture, in which several end users do not possess biometric information such as IoT devices.

## 3.7 Problem 3: traceable privacy-preserving in information sharing systems

Users' privacy is one of the most important challenges to address in both cloud and fog computing architectures. Indeed, data externalization in the fog or cloud servers might expose users' personal information to leakage threat. It is true that personal information leakage issue can be solved using cryptography, but preserving privacy is not limited in exposing users' identities or some of their private information to the public. It also concerns the detection of users' behavior pattern, activity tracking, interests and preference detection, etc. In fact, selling this kind of information to companies, interested in targeted advertising for example, may be much more useful for service providers than revealing users' identities.

To deal with these issues, data owners usually tend to anonymization techniques such as k-anonymity [Sweeney, 2002], periodical keys generation, pseudonyms-based authentication and group signatures, to avoid any kind of linkability to the users. However, in some applications, such as public information sharing applications, anonymization has a crucial drawback which is the lack of traceability. Indeed, users could misuse the system anonymity feature and start sharing false information, assault other users, etc. To illustrate this situation, let us consider the example of connected vehicles sharing traffic information through fog computing servers. Suppose that in order to preserve the privacy of the vehicle owners, traffic information has been anonymized to prevent fog servers from tracking the origin of information and find out where the vehicle owners are headed. Nevertheless, some malicious vehicle owners could share false information, for instance, saying that there have been a car incident somewhere. Due to this alert, the police may intervene, but they will discover that it was a false alert. However, because of anonymization, the police cannot trace the origin of this false alert in order to punish the malicious user. Under the light of the previous example, it is clear that full anonymity without any traceability mechanism can be a serious issue, especially in untrustworthy architectures such as fog computing.

Therefore, the challenging problem can be stated as follows: given a network of communicating entities that share public information through a computing architecture such as cloud or fog, how can we preserve the communicating entities'

privacy? Beside privacy-preserving service, how can we ensure that one trusted member of the network could trace any malicious entity, in case of abuse or anomaly detection?

### 3.7.1 Related work

There have been several proposals which addressed the privacy issue in the literature. Most of these proposals consider applications that can operate in fog or cloud computing architectures.

Liu et al. [Liu et al., 2012] proposed an anonymous payment system with privacy protection support. Their work provided the mechanisms to enhance location privacy of electric vehicles. Nicanfar et al. [Nicanfar et al., 2013] proposed a robust privacy-preserving authentication scheme for communication between the electric vehicles and power stations. Rottondi et al. proposed a security infrastructure for privacy-friendly V2G interactions [Rottondi et al., 2014a] [Rottondi et al., 2014b]. These previous proposals preserve privacy, but they did not provide any traceability service that allows to identify misbehaving entities.

In [Wang et al., 2015], the authors proposed a traceable privacy-preserving communication scheme in smart grids. In this scheme, there are three main components, the local aggregators (LAG), the central aggregators (CAG) and the electric vehicles. The vehicles use pseudonyms to hide their private information nearby the local aggregator. However, before formulating any request to the LAG, the vehicles need to contact the CAG in order to get its signature. Aslam et al. [Aslam and Zou, 2009] proposed a Distributed certificate architecture for VANETs. Each vehicle in this scheme has a temporary pseudonym that is valid in a specific area during a specific period. The vehicles can get these pseudonyms nearby components known as payment providers. However, the vehicles use the same pseudonym is a specific area, thus, they can easily be traced in this area. Moreover, since the payment providers generate pseudonyms, they will be able to trace the vehicles and violate their privacy. Salem et al. [Salem et al., 2010] proposed a non-interactive authentication scheme providing privacy among drivers in Vehicle-to-Vehicle Networks. In this solution, drivers are assembled in V2V communication groups. Each driver gets a pair of keys (public and private) from a trusted third party (TTP). Group members could frequently change their own set of public keys, and thus they ensure their privacy. Note that group members generate the new set of public keys without requiring a control from the TTP. However, to trace the drivers in case of misbehavior, the TTP need to try each private key stored in its database until it finds a match with the malicious driver's public key.

Liu et al. [Liu et al., 2013] proposed Mona, a multi-owner data sharing solution for dynamic groups in the Cloud. Both anonymity and traceability are well supported in this scheme. However, as long as the group manager did not verify the data signature, the cloud cannot make it available for group members. Shen et al. [Shen et al., 2018] proposed an anonymous and traceable group data sharing and storage scheme in the cloud which is similar to Mona. In this scheme, a group manager defines a group signature that is used to achieve anonymity. On the other hand, group members need to register nearby the group manager and receive a secret key. When a user wants to share data into the cloud, he first signs the data using its secret key and sends it to the group manager. The group manager verifies the signature and then replaces it with the group signature. Finally, the data will be uploaded to the Cloud. This scheme ensures both anonymity and traceability, but the group members need to pass through the group manager at any data-sharing event.

## 3.8 Conclusion

In this chapter, we have presented an overview on security issues in cloud and fog computing. We have started by ranging the main cloud computing issues discussed in the literature under different contexts (Data related issues, virtualization, identity and access management, malicious insider issue, software/internet technology issues, privacy issues). After that, we have presented the main challenges resulting from these issues. Then, we have introduced the first problem that we have treated during this thesis (the revocation in attribute-based access control systems) and its related works. Later on, we have presented the additional challenges that need to be addressed in fog computing paradigm. Under the light of these additional challenges, we have introduced the second problem that we have addressed in this thesis (mutual authentication in fog computing architecture) and its related works. Finally, we have presented a third problem that fits with both cloud and fog paradigms, and which addresses privacy-preserving in data sharing systems along with its related works.

# *Revocable attribute-based access control system*

## 4.1 Introduction

Multi-authority attribute-based encryption is an encryption method which provides a distributed, flexible and fine-grained access control in untrustworthy environments. However, this method suffers from some shortcoming as revocation which is one of its major challenges. The revocation consists of banishing users from the system or some of their attributes to prevent them from getting access to the data. In literature, the most known solutions, as time-based solutions and proxy solutions, suggest to attribute an expiration time to users' keys or to naively rely on a semi-trusted proxy to revoke users. In the time-based solutions, the revocation is not immediate and the revoked users might continue to access the data until the next key regeneration phase, while proxy-based solutions do not achieve fine-grained access and the users cannot get access if the proxy goes offline.

As far as we know, all existing solutions consider a single data-sharing domain (public) where all the users mutually share their data. However, in some situations, a user may want to share its data only with a specific group of users. This introduces a new data-sharing domain called the personal domain. The revocation in the public domain is managed by the authority and once a user is revoked, he will lose its access right in all data-sharing domains. But, the revocation in the personal domain should not affect the access right of the revoked user in the public domain. Thus, a new challenge in terms of revocation introduces it self as: how can we develop a new revocation level in which the data owner revokes the access of other users only in its personal domain, while these revoked users can continue to access the shared data in other domains?

In this chapter, we propose a fine-grained access control scheme with efficient attributes and users revocation. Based on the strength of secret sharing method [Shamir, 1979] in group management and Multi-authority CP-ABE [Lewko and Waters, 2011], our solution can be adapted to both centralized and fully

distributed data-sharing architectures and provides the possibility to share data in both public and personal domain, which is not the case in existing solutions. In addition, we provide through experimentation an advanced performance evaluation of our solution in terms of encryption/decryption and revocation computational cost. Our experimental results show that our solution does not affect the performance of the native decentralized attribute-based encryption and provides better results compared to existing solutions.

Our solution is secure, scalable and offers the following advantages:

1. No re-keying process is needed due to the allocation of the revocation to the secret sharing method. Thus, when a revocation occurs, our scheme ensures that the revoked user cannot get the original ciphertext and fails in the decryption process.

2. Immediate revocation of the users by changing the secret of the attributes' groups in such a way that only the authorized users could discover the new secret.

3. Low computation cost in the reconstruction of the attributes' secrets.

4. Flexible in case of users' joining and leaving the attribute groups.

5. The possibility to share data in a personal domain. Therefore, the data owner shares its data on an external server and controls the revocation as well.

6. The possibility to share data and manage the revocation in a fully distributed data exchange architecture, without introducing any new components in the architecture.

The remaining of this chapter is organized as follows. In Section 4.2, we present our solution. Then, we discuss its security in Section 4.4. We provide an application of our solution to evaluate its performance in Section 4.5. Finally, we conclude in Section 4.7.

## 4.2 Our solution

In this section, we present our solution which allows to perform immediate and efficient revocation in both attributes and users' levels. Using the secret sharing method, we propose a new revocation solution in Multi-authority CP-ABE access control model. Our solution does not require any key redistribution (when some changes occur in the users' attributes) to perform a revocation.

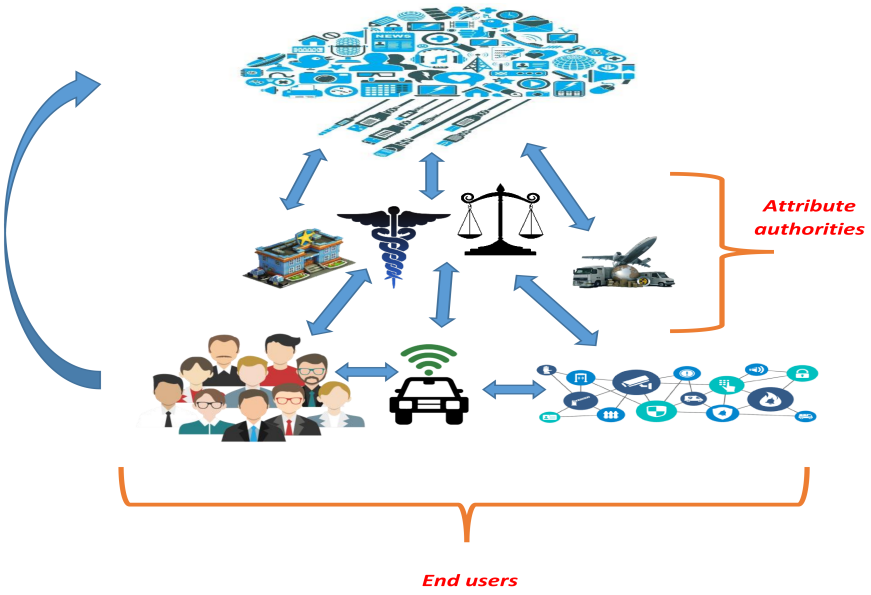



Figure 4.1 – Our architecture

We first present the considered architecture and its security requirements. Then, we introduce our secret sharing approach and show how we use it to manage the dynamic of the users in a general way, and the revocation in particular.

In our solution, we consider a data outsourcing architecture (Figure 6.1) composed of the following components:

- **Multiple authorities** which are responsible for managing a set of attributes and issuing attribute keys as well. These authorities do not need to coordinate or even be aware of each other.

- A set of **end users** such as connected vehicles, IoT devices or any person which has any interest in sharing data with the others.

- **Data externalization server** such as Cloud for data storage and sharing, in case where the entities share data in a centralized way.

**Security requirements:**

Our solution ensures the following security requirements:

1. Confidentiality: only authorized users should have access to the data, but the revoked users should not have the access. In addition, the proposed scheme should be adaptable to both backward and forward secrecy properties. Thus, our proposal needs to provide the possibility to manage the following situations:

- Backward secrecy: any user who comes to hold an attribute, (that satisfies the access policy), should be prevented from accessing the plaintext of the data exchanged before he holds the attribute.
- Forward secrecy: any user who comes to lose an attribute should be prevented from accessing the plaintext of the subsequent data exchanged after he loses the attribute, unless the other valid attributes that he is holding satisfy the access policy.

2. Collusion-resistance: the system must deal with situations where unauthorized users, who do not possess enough attributes, try to combine their keys in order to decrypt the data.

In what follows, we present some concepts that we will use in this chapter:

1. **User's share:** is a value assigned to each user in the group. This number is computed after the initialization phase.
2. **Complementary share:** is a value which is stored with the shared data. This number is combined with the specified user's share to reconstruct the secret of a specific attribute.
3. **public data-sharing domain:** is a data-sharing domain in which all the users in the system mutually share data. To successfully decrypt the data in the public domain, one needs to possess valid attributes which were not revoked by the authorithy and which verify the access policy.
4. **personal data-sharing domain:** is a data-sharing domain in which a user shares its data with a specific set of other users. Thus, to succesfully decrypt the shared data, one needs to be part of the data-sharing group and also possesses attributes which satisfy the access policy.

### 4.2.1 Multi-authority access control model for centralized data-sharing

In the system that we consider, each entity is defined according to a set of attributes that can be issued from several trusted authorities, called attribute authorities and denoted $AA$. Each attribute authority $AA_i$ generates private keys for each entity which is under its control. To ensure the revocation using our secret sharing method, each authority $AA_i$ generates a secret for each attribute $A_k$ that it manages. Then, it gives one share of that secret to each entity which possesses that attribute, while the other share is uploaded with the data.

In our solution, an entity encrypts the data using an attribute access policy and shares it in a public domain. The users could decrypt the shared data if they possess valid attributes which satisfy the access policy.

In this data-sharing scope, the authority is responsible for attributes revocation. Moreover, an entity could also create a personal data-sharing domain and shares the data only with a specific set of users. In this situation, only the data owner should be responsible for the revocation.

The revocation in the authority level means that the entity loses its attribute in all data-sharing domains (public and personal), while the revocation in the personal domain do not affect the access rights of the revoked entity in the public domain or any other personal domain.

### 4.2.2 Threat model

In our solution, we establish a group key for each attribute in the system. These group keys are used to prevent any unauthorized access to the ciphertexts, without performing the heavy key generation process. Therefore, in order to break our security scheme, the adversary aims to recover the group keys. In our architecture, the authority is assumed to be trusted, thus, the adversary can be either the cloud server or an end user. As it is usually assumed in the literature, the cloud is considered to be honest but curious. Therefore, it follows the protocol, but it will try to find out the group keys using the complementary shares stored with the data. We express the attack model for the cloud server through an indistinguishability game as follows:

Consider an adversary who is not in possession of a secret random value $R$, but he has the possibility to form two messages with the same length and sends them to a random oracle.

The oracle gives him back one of the two messages blinded with the secret value $R$. We note that the oracle chooses randomly one of the two received messages.

The scheme is considered secure if the adversary has a hard time to tell which one of the two messages was blinded with $R$.

**Challenge:** the adversary submits two numbers $S_0$ and $S_1$ where $S_0, S_1 \in Z_p^*$ to the random oracle. The oracle randomly flips a coin $b \in \{0, 1\}$ and selects uniformly at random from $Z_p$ a random value $R$, then it returns $a = S_b/R$ to the adversary.

**Guess:** the adversary outputs a guess $b'$ of $b$.

The advantage of the adversary in this game is expressed as:

$$Pr[b' = b] - 1/2$$

On the other hand, unauthorized end users may also try to recover the group key by combining their shares with other unauthorized users' shares. We describe the adversarial model in this situation as follows:

Consider an adversary who possesses a user share, and requests from the random oracle as much user shares as needed. The random oracle possesses a polynomial $P$ and $n$ predefined points. It also generates a new random point $p_i$ through $P$ each time that the adversary requests a new user share. After that, the oracle generates a user share using the new point $p_i$ and the $n$ predefined points and sends it to the adversary.

The adversary wins the game if after $k$ requests, he will succeed to recover the polynomial $P$, using the received shares, with a non negligible probability.

### 4.2.3 Revocation in the authority level

In what follows, we present the main idea of our revocation scheme for the authority level of data supervision along with its access control approach as:

- The authority $AAi$ chooses a secret for each attribute that it manages. Then, it computes a user and a complementary share for each entity that possesses that attribute. Next, it sends a share to each user and uploads the complementary shares into the externalization server.

- When an entity decides to upload the data, it defines an access policy to this data and encrypts it using this policy, then, it sends the resulting ciphertext to the externalization server.

- Next, the data owner sends the set of the leaf nodes defined in its access policy to the authority responsible for each attribute in that set. The corresponding authority launches a symmetric encryption on some parts, related to the attributes, of the uploaded ciphertext. We recall that the re-encryption process uses the secrets defined for the attributes in the received set.

- When an entity requests the data, the externalization server replies by sending the ciphertext and the complementary shares specified for that entity. First, the data requester uses his share and the complementary shares provided by the externalization server to reconstruct each attribute secret. The purpose of this phase is the decryption of the ciphertext pieces, which were encrypted using the secrets described above. Finally, the entity continues the decryption process as it was described on Multi-authority CP-ABE.

When a revocation occurs, the authority generates a new secret for the revoked attribute, and computes a new complementary share for each non revoked entity and sends them to the externalisation server.

This measurement allows authorized entities to reconstruct the new secret of the revoked attribute, using their share and the updated complementary share. However, the revoked entity can only retrieve the old secret of the attribute because its complementary part was not updated. So, it will not be able to succesfully decrypt the data.

**Construction:**

In what follows, we present the implementation of our secret sharing approach. Then, we present our centralized access control in general and the method which allows to realize the revocation in particular. In what remains, we consider the Cloud as a data externalization infrastructure for our revocation solution which works as follows:

#### 4.2.3.1 Initialization step

In this step, each attribute authority $AA_i$ defines a secret for each attribute that it manages, and computes a share for each user.

Since our solution is based on Shamir's secret sharing approach, the secrets selection and shares definition will be done as follows:

Given a set of attributes $S = \{A_{i,1}, A_{i,2}, \ldots . A_{i,n}\}$ managed by attribute authority $AA_i$. The algorithm chooses, for each attribute $A_{i,k}$ in S, a set of random values $a_i, i \in [1, n]$ from $Z_p$, and constructs a polynomial $P_{i,k}$ as:

$$P_{i,k}(x) = a_0 + a_1 x + a_2 x^2 + ..... + a_n x^n$$

Where $a_0$ is considered as the secret of attribute $A_{i,k}$ that will be used later in the re-encryption phase. We note that the degree of the polynomial does not depend on the number of users managed by each attribute authority. So, it is defined during the creation of each authority system.

After defining a polynomial for each attribute $k$, our algorithm chooses for each user $U_j$ a unique and random number $X_j$ which is used to generate a unique coordinates $(X_j, Y_j)$ through the polynomial $P_{i,k}$ defined above. These coordinates define a matrix called $MT$. On the other hand, the algorithm chooses $n$ (where $n$ is the degree of the polynomial) random numbers $X'_m$ totally different from those stored in $MT$. Then, it uses them to generate complementary coordinates $(X'_m, Y'_m)$ through the same polynomial $P_{i,k}$. These coordinates define a new matrix called $MC$. Later, both coordiantes defined in $MT$ and $MC$ are going to be combined in

order to compute users' and their complementary shares.

We note that each attribute authority maintains both matrixes $MC$ and $MT$ and sends only the complementary shares to the cloud storage server. So, the cloud server does not know anything neither about the polynomial chosen for each attribute, nor the coordinates generated through that polynomial and stored in the matrixes defined above.

#### 4.2.3.2 Add users

After defining the attribute secrets, each authority adds the set of users to the attribute groups and gives them shares that are used to reconstruct each attribute secret through secret sharing approach.

In our solution, we do not use native Shamir's secret sharing approach. This approach considers the coordinates $(X_j, Y_j)$ as user's share. Consequently, an update of these coordinates on the users' side will be required due to the changes that occur on the polynomial during the revocation.

Additionally, the native approach links the degree of the polynomial to the number of users to avoid the collusion problem. So, it is not scalable.

Sharing coordinates has many shortcomings such as scalability limit and updating issue. To address these challenges, our approach does not consider the coordinates stored in $MT$ as users' shares. Indeed, for each user our algorithm computes a user share using only the information that is not going to be changed when the secrets of the attributes change. We note that the value $X_j$ stored in $MT$ is a constant and unique value which is not affected by the changes of the polynomial. The same fact is applied to the values $X'$ stored in $MC$. These values are also unique and constant. So, if we compute a user share based on his unique value $X_j$ and the set of $X'$ values stored in $MC$, we deal with the shares update issue.

Consequently, the algorithm generates a specified share $L_{i,j}$ for user $U_j$ using the abscissa of the user's assigned point $X_j$ and the abscissa $X'$ of n chosen points from $MC$ through the following formula:

$$L_{i,j} = \prod_{m=1}^{n} \frac{-x'_m}{x_j - x'_m} \tag{4.1}$$

Where:

- $m$= index of a chosen point from $MC$.
- $j$= index of the user in $MT$.

We can notice that even if all the users in the system combine their shares, they will not reconstruct the secret since these shares are computed only with $X$ values and one needs a complete set of coordinates $(X, Y)$ to succesfully recontruct the secret. Therefore, the scalability problem of Shamir's scheme, which we cited above, is solved. Indeed, using shares computed with only $X$ values makes them useless if they are not combined with their specific complementary shares. So, it allows to unlink the polynomial degree from the number of users and solve the scalability issue.

Next, the authority assignes $L_{i,j}$ to user $U_j$ and stores it in $MT$ as user $U_j$'s share.

In order to provide a complementary share, which can be combined with the value $L_{i,j}$ calculated above, the algorithm computes, for each chosen point in $MC$, the $L_{i,m}$ value as follows:

$$L_{i,m} = \frac{-x_j}{x'_m - x_j} \times \prod_{z=1, z\neq m}^{n} \frac{-x'_z}{x'_m - x'_z} \tag{4.2}$$

Where:

- $m$= index of a chosen point from $MC$.
- $j$= index of the specified user in $MT$.
- $z$= index of a chosen point from $MC$ different from $m$.

After that, the algorithm computes for each user $U_j$, whose attributes are managed by attribute authority $AA_i$, the final complementary share using the $L_{i,m}$ and $L_{i,j}$ values calculated above and the second pair of the user $U_j$ coordinates $(Y_j)$ as follows:

$$CS_{i,j} = Y_j + \frac{\sum_{m=1}^{n} Y'_m \times L_{i,m}}{L_{i,j}} \tag{4.3}$$

Where:

- $m$= index of a chosen point from $MC$.
- $j$=index of the user's point.
- $Y_j$= ordinate of User i's point.
- $Y'_m$= ordinate of a chosen point from $MC$.

Finally, this complementary share is stored on the cloud server to be used to reconstruct the attribute secret.

#### 4.2.3.3 Data encryption

In our scheme, data is encrypted as follows:

First, the data owner runs the encryption algorithm defined in multi-authority ABE. After that, the secret of each attribute $x$ in the access policy will be used to re-encrypt the parts denoted by $C_{1,x}$ and $C_{3,x}$ in the resulting ciphertext. Therefore, the re-encrypted ciphertext is given by:

$$CT' = (C_0, E(\text{attribute x' s secret}, C_{1,x})$$
$$C_{2,x} = g_1^{r_x}, E(\text{attribute x' s secret}, C_{3,x}) \forall \text{ attribute } x)$$

Where:

- $x$ is an attribute from the access policy.
- $E$ is an encryption function which takes a symmetric key and the data to encrypt as parameters.

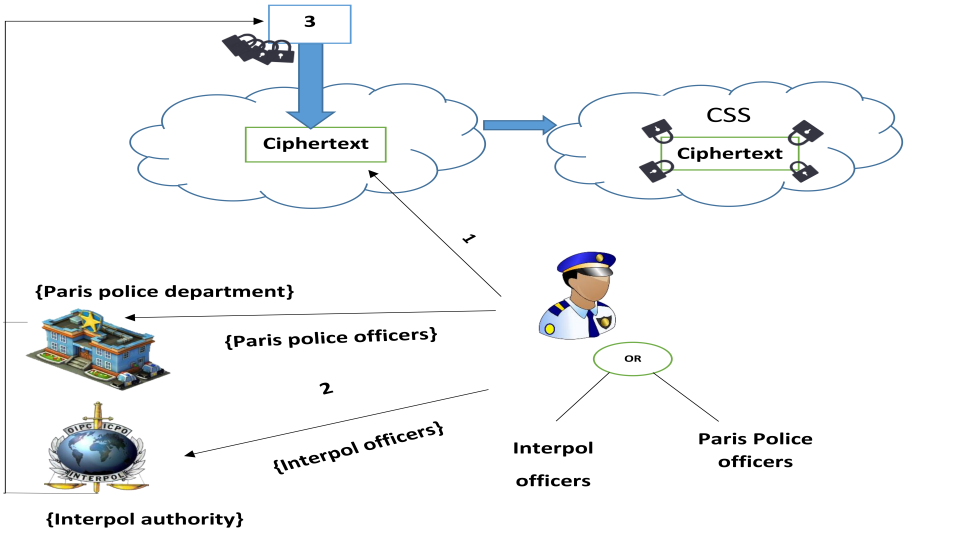
When a user wants to upload the data, he defines an access policy $P$ and encrypts its data, according to this policy, through multi-authority attribute-based encryption (Figure 4.2- step 1). Then, he sends the attributes defined in the leaf nodes of the access policy to the corresponding authorities (Figure 4.2- step 2). Finally, each authority (responsible for an attribute $x$ in $P$) re-encrypts the parts $C_{1,x}$ and $C_{1,3}$ of the uploaded ciphertext (Figure 4.2- step 3).

#### 4.2.3.4 Data decryption

The first step of the decryption algorithm is the reconstruction of the attribute secrets. We remind that each user $U_j$ possesses a secret share $L_{i,j}$ (eq.4.1). Furthermore, the complementary shares $CS_{i,j}$ (eq.4.3) specified for each attribute possessed by $U_j$ are stored in the Cloud server.

As shown in Figure 4.3, when a user requests the data, the Cloud storage server replies by sending the re-encrypted ciphertext and the complementary shares specified for that user. Once this information is received, the user reconstructs each attribute secret by multiplying his share with the specified complementary share:

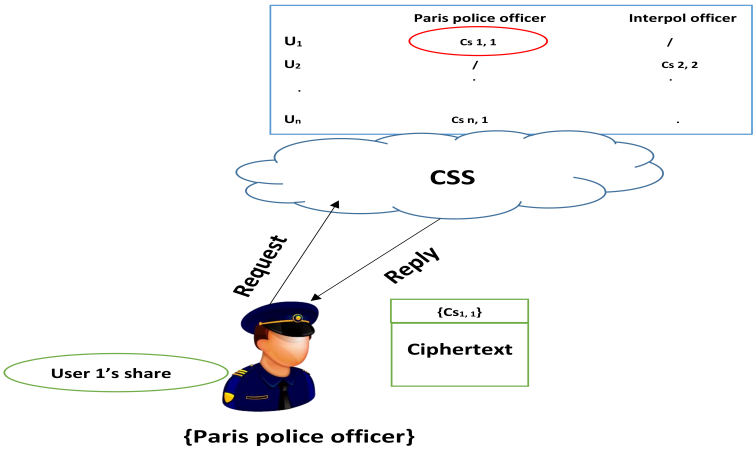
$$Attributesecret = L_{i,j} \times (Y_j + \frac{\sum_{m=1}^{n} Y'_m \times L_{i,m}}{L_{i,j}}) \tag{4.4}$$




1. Data owner encrypts the data under his access policy and uploads it on the CSS.
2. Data owner sends the leaf nodes of the access policy.
3. Re-encryption using leaf attributes' secrets.

Figure 4.2 – Data encryption and upload processes

Next, he performs a symmetric decryption on the parts of the ciphertext that were re-encrypted, using the reconstructed attribute secrets (Figure 4.3- step 1). Finally, he continues the decryption process exactly as it was described in Multi-authority CP-ABE scheme (Figure 4.3- step 2).




1. The user combines his share and the received complementary shares in order to decrypt the cipher-text's parts, which were encrypted using the attributes' secrets.
2. User 1 continues the decryption of the cipher-text using his attribute keys.

Figure 4.3 – Data download and decryption processes

#### 4.2.3.5 Attribute revocation

when the authority $AA_i$ decides to revoke an attribute $A_{i,k}$ from a user $U_j$, it runs the following actions:

1. Add the user to attribute $A_{i,k}$ revocation list (Algorithm 1, line 2).
2. Choose randomly from $Z_p$, a new secret by generating a new polynomial $P_{i,k}$ specific for the revoked attribute (Algorithm 1, line 3).
3. Generate new complementary points using the new polynomial(Algorithm 1, line 4).
4. Compute new complementary shares for each non-revoked user (Algorithm 1, lines $5-10$).
5. Update the complementary shares in the Cloud server (Algorithm 1, line 11).

**Algorithm 1** Revocation algorithm

```
1: global matrixes MC, MT
2: procedure REVOCATION(revocation_list RL,User_id, attribute A_i)
3:     List CS_list
4:     Add_revocation_list(RL,user_id,A_i)
5:     Generate_a_new polynomial (P_{i,k})
6:     Generate_new_complementary_points(MC)
7:     for each user U_i do
8:         if U_i not in RL then
9:             CS=Compute_CS(U_i_id,MC,MT)
10:            ADD_CS(CS_list,CS)
11:        end if
12:    end for
13:    Send_to_Cloud(CS_list)
14: end procedure
```

### 4.2.4 Revocation in the personal data-sharing domain

In our solution, we propose a new revocation level in which we allow the data owner to revoke other entities in its personal data-sharing domain. We note that it is useless to realize attribute revocation in that case, because the revocation in the personal domain targets the identity of the entities and not the attributes that they possess. Our solution for data sharing and revocation in the personal domain works as follows:

- The data owner sends a provisional secret along with a list of legitimate entities to the corresponding authority.

- The authority uses the received secret to compute complementary shares for each entity in the received list. The complementary shares are computed using each entity share as follows:

$$CS_i = \frac{S}{r_i \times L_i}$$

Where $r_i$ is a unique random number chosen from $Z_p$, and specified for each entity in the system. We note that each entity in the system knows its appropriate $r_i$ value. Unlike the complementary shares used for public data-sharing scope, we have introduced a random value $r_i$ to compute complementary shares in personal data-sharing scope. In fact, the complementary share in the public domain is computed using the user share $L_i$ and the secret of the attribute $S$. Since the secret $S$ is known only by a trusted authority, the confidentiality of the users share is ensured. However, in the personal domain, the data owner is the only entity that chooses the secret of the group. Consequently, our solution uses $r_i$ to preserve the confidentiality of the users share. Otherwise, if we use the same complemntary shares as public data-sharing domain, any malicious entity can violate the confidentiality of the users' shares by simply requesting the authority to provide complementary shares for data-sharing in its personal domain.

- Next, the data owner defines an access policy and runs Multi-authority ABE encryption phase. Then, he re-encrypts the part $C_0$ of the resulting ciphertext using a new secret $S_{new}$, while the authority re-encrypts the $C_{1,x}$ and $C_{3,x}$ parts using the secret of each attribute $x$ as follows:

$$CT' = (E(S_{new}, C_0), E(\text{attribute x' s secret}, C_{1,x})$$
$$C_{2,x} = g_1^{r_x}, E(\text{attribute x' s secret}, C_{3,x}) \forall \text{attribute x})$$

Where:

  - $\diamond$ $x$ is an attribute from the access policy.
  - $\diamond$ $E$ is an encryption function which takes a symmetric key and the data to encrypt as parameters.

- After that, the data owner modifies the complementary shares provided by the authority using $S_{new}$ as follows:

$$CS_{inew} = CS_{iold} \times \frac{S_{new}}{S_{old}} \tag{4.5}$$

- When an entity requests the data stored in the outsourcing server, the server replies by sending the encrypted data and the complementary shares specific for that entity. First, the data requester combines his $r_i$ value, his share, and the complementary one to reconstruct the sharing-group secret as follows:

$$S = r_i \times L_i \times CS_i$$

  Then, he performs a symetric decryption on the $C_0$ part which was re-encrypted by the data owner.

  After that, the data requester continues the decryption as it was descibed in data decryption phase above (subsection 4.2.4).

- When the data owner decides to revoke an entity from its personal domain, he defines a new secret and updates the complementary shares provided by the authority for each non-revoked entity, exactly as it was shown in (eq.4.5).

## 4.3 Multi-authority access control model for fully distributed data-sharing

In addition to centralized data-sharing model, sometimes the users might want to share the data in a fully distributed manner without using any central server. This data-sharing approach is more challenging due to the untrustworthy nature of the architecture. Indeed, in this approach, the data owner will broadcast its data into the network and should prevent unauthorized users from getting access to the data content. Moreover, he should also be able to revoke any user in this data-sharing group.

With some changes in the complementary shares of our secret sharing scheme, we propose a solution that also manages access control models in fully distributed data-sharing architecture. Our proposal works as follows:

- The data owner defines a cyclic group $G$ of prime order $p$ with a generator $g$. Then, he chooses a provisional secret $S \in Z_p$, and computes a provisional key $g^S$.

Finally, he encrypts $g^S$ under an access policy and broadcasts it in the network. This broadcast operation aims to form the data-sharing group, i.e. detecting all the users who are able to satisfy the access policy in the network. Figure 4.4, describes the operations realized by the data owner in the first step of the fully distributed data-sharing.

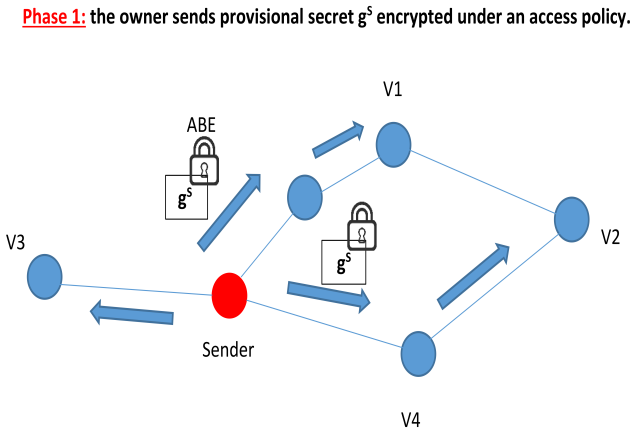



Figure 4.4 – Reconstruction of data-sharing group

- Each node, willing to join the data-sharing group, needs to successfully decrypt the received ciphertext. Once done, he should compute a complementary share as follows:

$$CS_i = (g^S)^{1/Li} = g^{\frac{S}{L_i}}$$

Finally, he broadcasts its complementary share in the network. We note that sharing the complementary share in that form does not allow other users to recover the $L_i$ value due to the hardness of discrete logarithm problem in cyclic groups. Thus, this complementary share is useful only when it is combined with its specific user share, otherwise it becomes useless. Figure 4.5, shows how the users could join the data-sharing space.

- Next, for the same access policy defined above, the data owner encrypts the data using a new secret $g^{S_{new}}$ as a symetric key, and updates the received complementary shares as follows:

$$CS'_i = CS_i^{\frac{S_{new}}{S_{old}}} = (g^{\frac{S_{old}}{Li}})^{\frac{S_{new}}{S_{old}}} = g^{\frac{S_{new}}{Li}} \tag{4.6}$$

Figure 4.6, shows the data-sharing process.

- When a the data owner decides to revoke a group member, it changes the complementary shares of non-revoked entities, linked to the appropriate

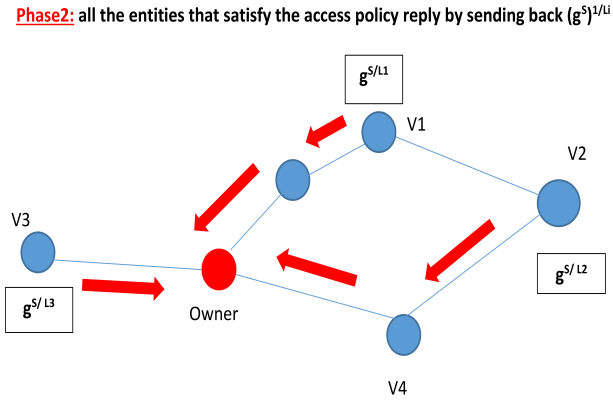



Figure 4.5 – Joining data-sharing group

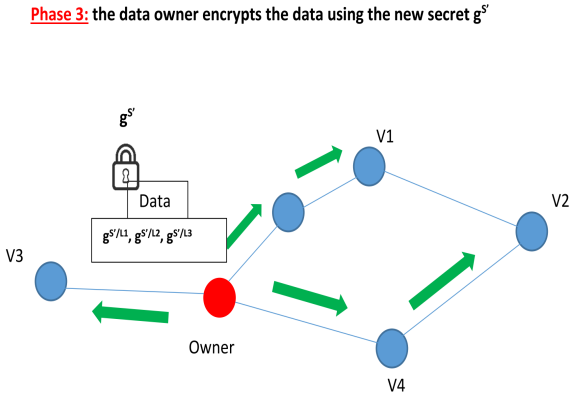



Figure 4.6 – Data sharing phase

ciphertext, as shown in (eq.4.6). Figure 4.7, summarizes the revocation of a group member.

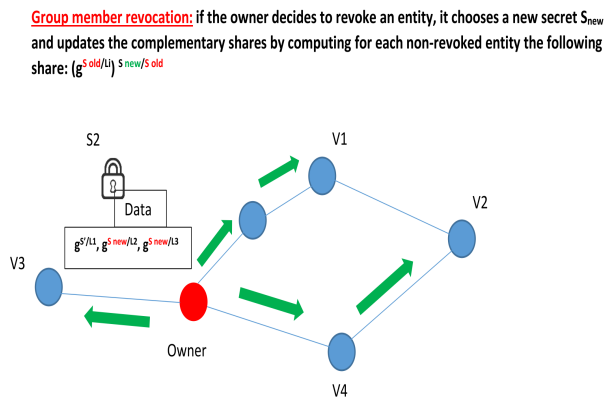



Figure 4.7 – Group members' revocation

We note that we do not consider the possible authentication and integrity issues that this architecture can face off.

## 4.4 Security analysis

In this section, we verify the security of the proposed secret sharing process.

**Definition 1.** *the discrete logarithm problem is defined as:*

*Given $g$, $h \in G$, find an $x$ such that $g^x = h$. The difficulty of this problem depends on the group $G$:*

- *Very easy: polynomial time algorithm, e.g. (ZN , +)*
- *Hard: sub-exponential time algorithm, e.g. (Zp, ×).*
- *Very hard: exponential time algorithm, e.g. elliptic curve groups.*

**Definition 2.** *an internal composition law on a set $E$ is a mapping of $E \times E$ into $E$.*

**Definition 3.** *a set $G$ is called a group if it has an internal law $T$ having the three following properties:*

- *it is associative: $(xTy)Tz = xT(yTz)$*
- *it has a unit e: $eTx = xTe = x$*
- *every element $x$ of $G$ has an inverse $x_0$ : $xTx_0 = x_0Tx = e$*

*A composition law with these properties is called a group law. If, further, the law $T$ is commutative $(xTy = yTx)$, the group is called commutative or Abelian.*

**Definition 4.** *a ring is a set $A$ endowed with two internal composition laws, the first being that of an Abelian group, the second being associative, and distributive with respect to the first. If we write the first law additively and the second multiplicatively, then:*

- *First Law:*

$$\begin{aligned} (x+y)+z &= x+(y+z) \\ x+e &= e+x = x \\ x+(-x) &= e \\ x+y &= y+x \end{aligned}$$

- *Second Law:*

$$(xy)z = x(yz)$$

- *Distributive Law:*

$$(x + y)z = xz + yz$$
$$z(x + y) = zx + zy$$

*If the second law is also commutative* $(xy = yx)$, *A is called a commutative ring.*

*If the second law has a unit element e such as* $(xe = ex = x)$, *it is called a unit of A and A is called a ring with unit.*

**Definition 5.** *let K be a ring and e the unit for the first law (the Abelian group law); let* $K^*$ *be the set of elements of K other than e. If the second law on K is a group law on* $K^*$, *K is called a field.*

**Theorem 1.** *the ring* $Zn = Z/nZ$ *is a field only if n is a prime.*

### 4.4.1 Data Confidentiality

Our solution is based on two main approaches, Multi-authority CP-ABE and Shamir's secret sharing scheme, in which data confidentiality has been proven on their original papers [Lewko and Waters, 2011, Shamir, 1979]. However, in our solution, we do not use native Shamir's secret sharing scheme. So, what remains is to prove that using this approach as it was described in section 4.2 is still secure.

Our proof involves both centralized and fully distributed data-sharing. First, we prove that the users' shares do not disclose any information about the secrets of the attributes. After that, based on the hardness of the discrete logarithm problem, we show that the complementary shares are secure in the fully distributed data-sharing architecture. Finally, we prove through an indistinguishability game that our complementary shares are secure as well in the centralized data-sharing domain.

#### 4.4.1.1 users' share security

The security proof of our scheme relies on one of Shamir's secret sharing scheme properties, which is the perfect secrecy property.

By definition, this property means that a polynomial $P$ of degree $t-1$ is uniquely determined by any $t$ shares calculated through $P$, and hence the secret $a_0$ can be computed. However, given $t-1$ or fewer shares, the secret can be any element in the field $Z_p$, and thus those shares do not supply any further information regarding the secret.

We recall that the shares in Shamir's scheme are the set of pairs $(X_i, Y_i = P(X_i))$ where $X_i, Y_i \in Z_p^*$. However, in our scheme the user' share is calculated

using only the values $X$ and $X'$ stored on the matrix $MT$ and $MC$.

If we compare our users' shares with the shares used in Shamir's approach, we notice that our shares are computed with $n$ incomplete Shamir's scheme shares ($L_{i,j}$ contains only $X_i$ values, but none of the $Y_i$ values) and according to the secrecy property, our users' shares reveal nothing about the secret since the secret construction uses both $(X_i, Y_i)$ values.

In our fully distributed data-sharing architecture, the complemtary shares are published as $CS_{i,j} = g^{S/L_{i,j}}$, where $S$ is the secret chosen by the data owner, $L_{i,j}$ is the user's share and $g$ is the generator of a cyclic group $G$ of prime order $p$.

#### 4.4.1.2 Complementary shares security

The security proof of the complementary shares depends on the data-sharing model:

A) **In the fully distributed data-sharing:** due to the hardness of the discrete logarithm problem in cyclic groups (definition 1), the users' shares and the group secret are protected. Indeed, the data owner cannot use the complementary shares to recover the users' shares $L_{i,j}$. On the other hand, once the user is revoked, he cannot recover the group secrets as well because the shares of non revoked users will be updated as $CS_{i,m} = g^{S'/L_{i,m}}$ (where $m \neq j$ and $S'$ is the new secret of the group), while the revoked user's share do not change and remains as $CS_{i,j} = g^{S/L_{i,j}}$.

B) **In the centralized data-sharing:** we remind that in our scheme, the algorithm chooses $n$ randomly coefficient $a_i$, where $\forall i \in N, a_i \in Z_p$. Then it constructs a polynomial $P_i$ as follows:

$$P_i(x) = a_0 + a_1 x + a_2 x^2 + ..... + a_{n-1} x^{n-1}$$

Given the secret $S_i = P_i(0)$ of an attribut $i$, the complementary shares are computed as follows:

$$CS_{i,j} = \frac{S_i}{L_{i,j}}$$

Where:

- $S_i$= is the secret of the attribute $i$
- $L_{i,j}$= user $j$'s share

We prove that $CS_{i,j}$ values stored in the cloud, either on the centralized public or the personal domain, does not disclose any information about the secret above

is provided through an indistinguishability game. The main idea of this game is to consider an adversary who is not in possession of a secret random value $R$, but he has the possibility to form two messages with the same length and sends them to a random oracle.

The oracle gives him back one of the two messages blinded with the secret value $R$. We note that the oracle chooses randomly one of the two received messages.

The scheme is considered secure if the adversary has a hard time to tell which one of the two messages was blinded with $R$. We call the adversary's advantage in that kind of game the probability of its success to break the scheme. It is expressed as :

$$Adv(A) = 2\Delta Pr[GuessA = true] - 1$$

.

**Challenge:** the adversary submits two numbers $S_0$ and $S_1$ where $S_0, S_1 \in Z_p^*$ to the random oracle. The oracle randomly flips a coin $b \in \{0, 1\}$ and selectes uniformly at random from $Z_p$ a random value $R$, then it returns $a = S_b/R$ to the adversary.

**Guess:** the adversary outputs a guess $b'$ of $b$.

The advantage of the adversary in this game is expressed as:

$$Pr[b' = b] - 1/2$$

.

Giving the right answer based only on the value of "$a$" means that the adversary is able to determine a unique operation $S_b/R$ which results "$a$". However, $\forall a, S_0, S_1 \in Z_p^*, \exists R_1, R_2 \in Z_p^*$ where:

$a = S_0/R_1$ and $a = S_1/R_2$. So, there are two different ways to compute the same value "$a$" given "$S_0, S_1$".

**Conclusion:** considering that the secret values $R$ are uniformaly chosen at random in $Z_p^*$, we can conclude that given the values "$a, S_0, S_1$", the probability that the oracle chooses $R_1$ or $R_2$ is:

$$Pr[R = R_1] = Pr[R = R_2] = 1/2$$

.

Consequently, we can say nothing about how the value of "$a$" has been computed, since it can be calculated with two different ways with the same probability. Therefore, knowing the value "$a$" reveals nothing about which pair of values $(S_0, R_1)$ or $(S_2, R_2)$ has been chosen by the oracle. According to that fact, the probability that the adversary chooses the right answer remains $Pr[b' = b] = 1/2$ for any pair

$(S_0, S_1)$ sent to the oracle.

The purpose of the indistinguishability game is to prove that even if the adversary chooses the attribute secrets, he cannot achieve a reasonable advantage in recovering the $R$ value.

This situation matches with our personal data-sharing domain since the users are allowed to choose the secrets in their personal domain. In addition, it proves also that secrets' recovery is even harder in the case where the adversary does not possess any information about the secrets at all, which matches with our public data-sharing domain where the cloud knows nothing about the attributes secrets.

### 4.4.2 Collusion resistance

The collusion resistance property consists of providing a protection against entities who do not possess an attribute. We note that we do not consider situations where a legitimate user reconstruct the attribute secrets and provide them to an illegitimate one.

In our solution, each entity holds one part of the attribute secret. This part is computed through a combination of multiple secret information ($X_j$ and $X'$ values) known only by the authority. On the other hand, the secret reconstruction requires a combination of a set of coordinates $(X, Y)$. Thus, we can clearly notice that the users' shares cannot recover the attribute secrets even if all the users combine their parts together. Indeed, the parts in possession of the users do not contain the $Y$ values required in the secret reconstruction. Therefore, these parts become useless if we do not combine them with their complements available on the storage server or attached to the data, in the case of fully distributed data-sharing. In addition, the entities can only use their specific complementary shares to recover the correct attribute secret. Otherwise, the combination of incompatible shares will result an incorrect secret. If the Cloud is dishonest, each set of complementary shares should be encrypted with the public key of its legitimate user to prevent any further collusion between the Cloud and end-users.

### 4.4.3 Forward secrecy

As the revoked users' complementary shares are not updated when they lose an attribute, they could reconstruct only the previous attribute's secret. So, they will not be able to decrypt the re-encrypted ciphertext's parts, since new secrets are going to be used in the following encryption operations.

### 4.4.4 Backward secrecy

The backward secrecy property is ensured by adding last data update date to the access policy. To do so, the date of ABE-key distribution is considered as an attribute that must be satisfied in the access policy (using "And" gate). Hence, a new joining member can only have the access if his key's distribution date is less than the last data update date.

## 4.5 Application and performance evaluation

In this section, we apply our secure data-sharing scheme on connected vehicle applications. Then, we evaluate its performance on a real connected vehicles use case [Pol, , dub, ].

Nowadays, connected vehicles have taken more intention in both academia and industry due to its wide application spectrum, such as data-sharing, cooperative collision warning, improved rescue, road obstacle detection, etc. It is predicted that around 200 million connected vehicles will be on the road in 2020 [rob, ]. It is true that these applications open a huge amount of opportunities, but also introduce several challenges [con, , Lu et al., 2014]. Security is one of the major concerns in connected vehicles applications [Rivas et al., 2011, Whaiduzzaman et al., 2014]. Indeed, ensuring the safety of exchanged data through enforced security protocols is an important step in the establishment of theses applications.

In the connected vehicles data-sharing applications, we can distinguish three principal components in the architecture: the connected vehicles, the authorities that might be organizations or the person who owns the connected car, and finally, a set of entities "persons or connected objects", which might have interest for the exchanged data.

We also might include data storage infrastructures, such as Cloud, as a part of the architecture in the case where the data is shared in a centralized way.

We recall that our solution operates on an attribute-based system. Therefore, each entity in the architecture will be characterized by a set of attributes issued from one or multiple authorities. According to these attributes, the authority generates private keys to each entity which might have any interest for data-sharing. On the other hand, each authority defines secrets for each attributes and computes shares to each entity as shown in section 4.2, in order to allow the management of revocation.

The data-sharing in connected vehicle applications can take several directions:

### 4.5.1 Centralized data exchange

The first data-exchange model is the central one, in which the connected vehicles can exchange data with its authority, or share information with other vehicles through any service infrastructure, such as Cloud. For sake of illustration, we cite situations where a police department shares data with patrol vehicles, a taxi company which gives instructions to its connected taxi vehicles, etc.

We note that, the data exchange can also concern other entities which are neither connected vehicles nor owners of the vehicles. To clarify the idea, we can cite the example of a connected vehicle which share data about its technical state with a maintenance garage or any service terminal.

Figure 4.8, illustrates the sequence diagram of a centralized data-exchange application. First, an initialization step which aims to establish the system parameters and provides the users with the elements required for any further secure data-exchange is launched. After that, the users could perform data-exchanges as defined in subsections 4.2.3.3 and 4.2.3.4.

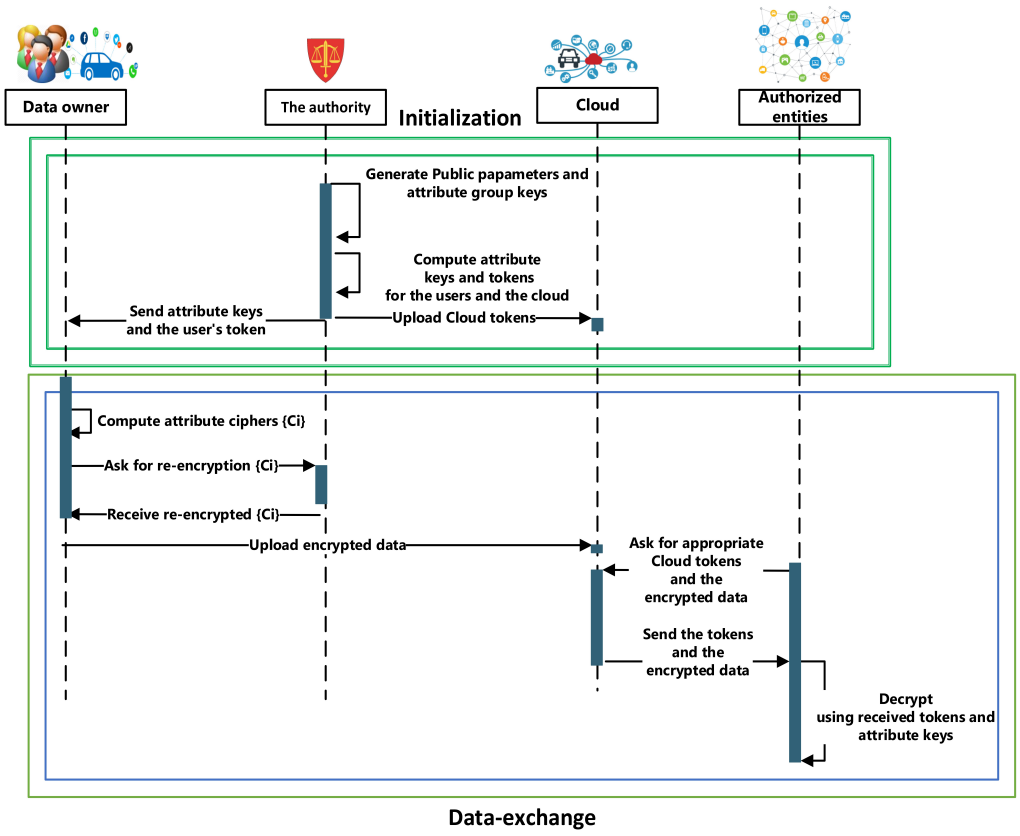



Figure 4.8 – centralized data-exchange

The revocation in centralized data-sharing model for connected vehicles applications can be performed in two different levels and for multiple reasons. Indeed, we

might have situations where a connected vehicle does not stay under the command of an authority due to several reasons such as "vehicle theft, breakdowns, etc.". Thus, the authority should perform an immediate revocation process in order to prevent any unauthorized access via the revoked vehicle. To manage such a situation, our centralized revocation scheme is more suitable.

On the other hand, we also might have situations where a connected vehicle decides to prevent an entity from getting access to the data, knowing that the revoked entity is still under the command of the authority. Therefore, we consider this sharing mode as a personal data-sharing and thus, our revocation scheme for the personal domain becomes more suitable.

### 4.5.2 Fully distributed data-exchange

In addition to the centralized data-sharing mode, we might have situations where the connected vehicles exchange data in a fully distributed manner. Indeed, if the connected vehicles are in the same area, for example, it will be more suitable and efficient to broadcast the data instead of using any third party for data-sharing. Thus, providing a convenient fully distributed data-exchange model is an important requirement for connected vehicles applications. Our fully distributed revocable solution meets perfectly these needs. Indeed, it does not just allow secure data exchange, it also permits the data owner to control the access and revoke any entity in a distributed way as well.

Figure 4.9, illustrates the sequence diagram of the fully distributed data-exchange application. First, the data owner forms the data-exchange group by defining a group key. Then, he broadcasts this key in the network. On the other hand, other users could join the exchange-group by sending back a token computed using the received key. Finally, the data-exchange step can start once the owner receives the tokens.

As we can notice, the flexibility and adaptability of our scheme to both centralized and fully distributed architectures, makes it strongly suitable to secure data exchange and ensure access control for connected vehicles applications.

## 4.6 Performance evaluation

To measure the performance of our solution and provide a comparison with other revocation methods, we implemented an access control application suitable for a data-exchange model of a real connected vehicle use case [Pol, , dub, ]. This use case consists of the autonomous police vehicles that the city of Dubai is set to introduce at the end of the year 2017. According to [dub, ], the cars are going to collect the

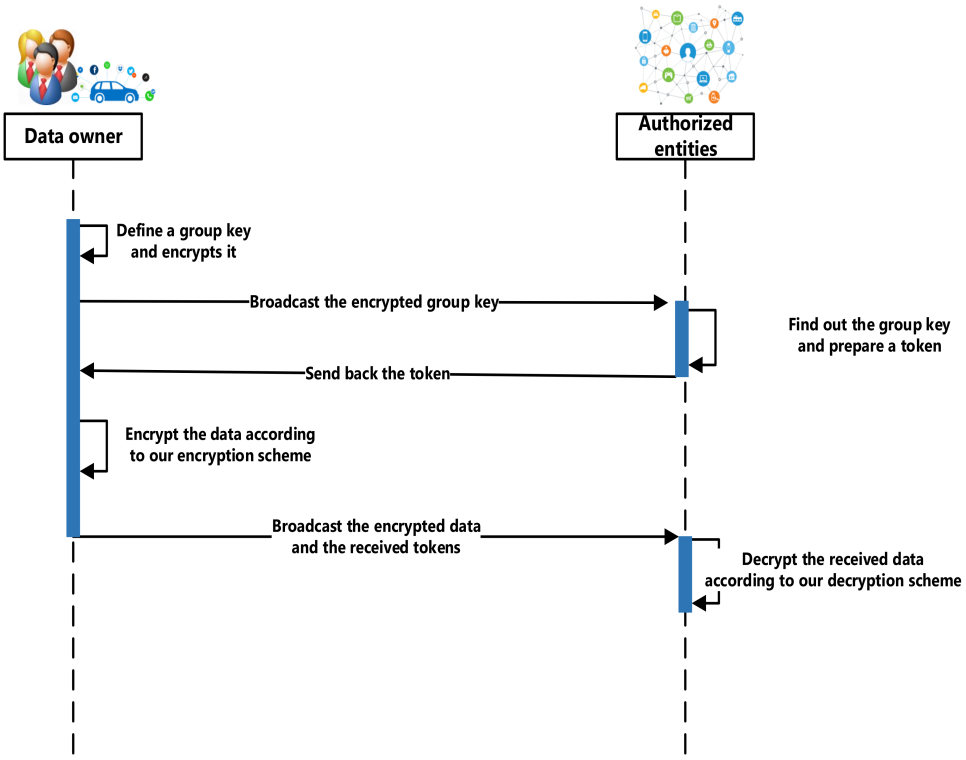



Figure 4.9 – Fully distributed data-exchange

data all over around the city in a 24hours/7days workload, and this collected data is directly sent to a command station.

The scenario that we propose to ensure access control with efficient revocation mechanism for this use case works as follows:

We implement the data-sharing module of each autonomous police car as a program which shares periodic reports to the command station employees through the Cloud. We note that in our implementation, we consider Dropbox [dro, ] as a cloud storage infrastructure.

In [Pol, ], there have not been any further details about the features of the data-sharing module in these cars except the fact that the command station will be able to supervise the images collected during their patrol. However, since our implemented data-sharing module performs encryption/decryption on a real data files and upload/download operations are also done on a real cloud platform [dro, ], we can say that it matches at least with a secure and standard data-sharing module that could be implemented for this use case.

To manage the 24hours/7days workload imposed by the police cars, we suppose that the command station employees work according to alternation system. Thus, we proposed to distribute the employees over three groups. Each group works one

Tableau 4.1 – Experimentation settings

| | |
|---|---|
| **Number of connected vehicles** | 50 |
| **Running time** | 10 minutes |
| **Reporting rate** | one report/3s |
| **size of report files** | {1..6} MB |
| **Number of police departement employees** | {1..1200} |
| **Maximum attributes/ key** | 20 |
| **Maximum nodes in the access policy** | 10 |

day and recovers two others. We divide the employees of the same group into three teams. Each team works eight consecutive hours.

To avoid any data leakage, which can be caused by the employees outside their work round, we proposed to perform a temporary revocation at each work team substitution. We recall that the revocation process must prevent each command station employee, who is outside its work round, to get access to the shared data.

Through the proposed scenario, we evaluate the computational cost of both encryption and decryption processes. We note that we compare these two processes through two access control versions: the first uses our access control solution while the second operates with the native attribute based scheme.

In addition, we provide an evaluation of our revocation solution and compares it with the time-based solutions.

The application was launched on an hp computer with 2.6 GHZ i7 and 16 GB of RAM. The experimentation settings are shown in table 4.1.

In our experimentation, each actor (vehicle, employee, authority) saves the period of time between the beginning and the end of any task (initialization, encryption/decryption and revocation) in a log file. At the end of the execution, we measure the computational time of each task as the average of all periods of time that have been recorded in its log file.

#### 4.6.0.1 Initialization cost

The initialization phase consists of defining the attributes secrets and adding the users to the attributes groups. In this phase, the authority computes the users and their complementary shares. Figure 4.10, shows the average computational cost of the initialization phase. To evaluate the performance of our scheme, we proposed to variate the number of the control station employees and the attributes which they possess as well. We note that we perform the attribute secret definition and the

Tableau 4.2 – Comparatif table of Attribute revocation methods for CP-ABE Systems

| **Revocation solutions** | Pirretti et al.,2010 | Yang et al., 2012 | Yang and Jia, | Zhong et al., 2018 | Ours |
|---|---|---|---|---|---|
| Computation in the authority side | $nb_{nu} \times nb_{att} \times exp$ | $nb_{nu} \times exp$ | $2 \times exp$ | $exp + mult$ | $nb_{nu} \times div$ |
| Computation in the user side | 0 | 0 | $mult$ | $mult$ | 0 |
| Key update | Yes | Yes | Yes | Yes | No |

$nb_{nu}$: the number of non revoked users. $nb_{att}$: the number of attributes contained in the secret key.
$exp/div/mult$: exponentiation/multiplication and division operations resp.

modular division operations on a finite field $Z_p$, where $p$ is a prime of twenty digits.

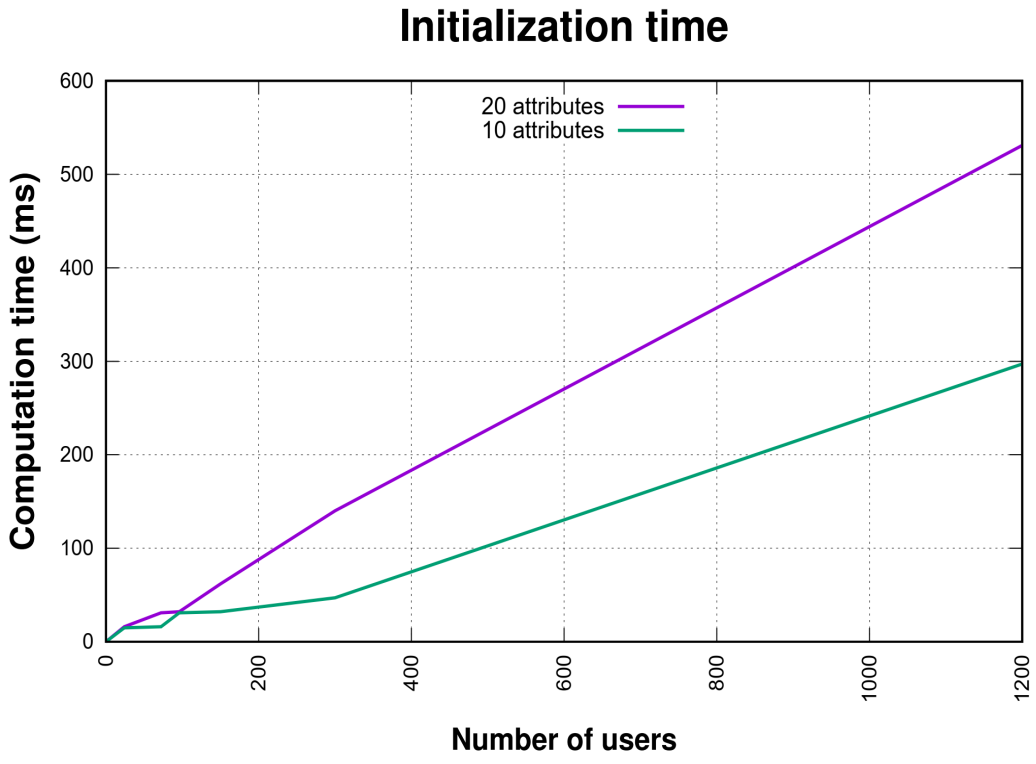



Figure 4.10 – Initialization time

As shown in figure 4.10, it is clear that the computational time of the initialization phase rises linearly with the number of the control station employees and the attributes that they possess as well, which is logical since in this step, the authority is only browsing the set of users and computing modular divisions for each user.

Tableau 4.3 – Revocation time

| Number of employees | 8 | 24 | 32 | 50 |
|---|---|---|---|---|
| Time-based solution [Yang et al,. 2012.](ms) | **7083** | **21554** | **29798** | **45027** |
| Our solution (ms) | **1211** | **1239** | **1380** | **1353** |

#### 4.6.0.2 Data encryption cost

We evaluate the encryption process on both our access control version and the native attribute-based encryption using multiple report files with different sizes. Figure 4.11, shows the average computational cost of both encryption versions using different access control policies not only in term of leaf nodes number in the access policy, but also in term of access policies complexity where we have to respect priorities in the access policy verification phase.

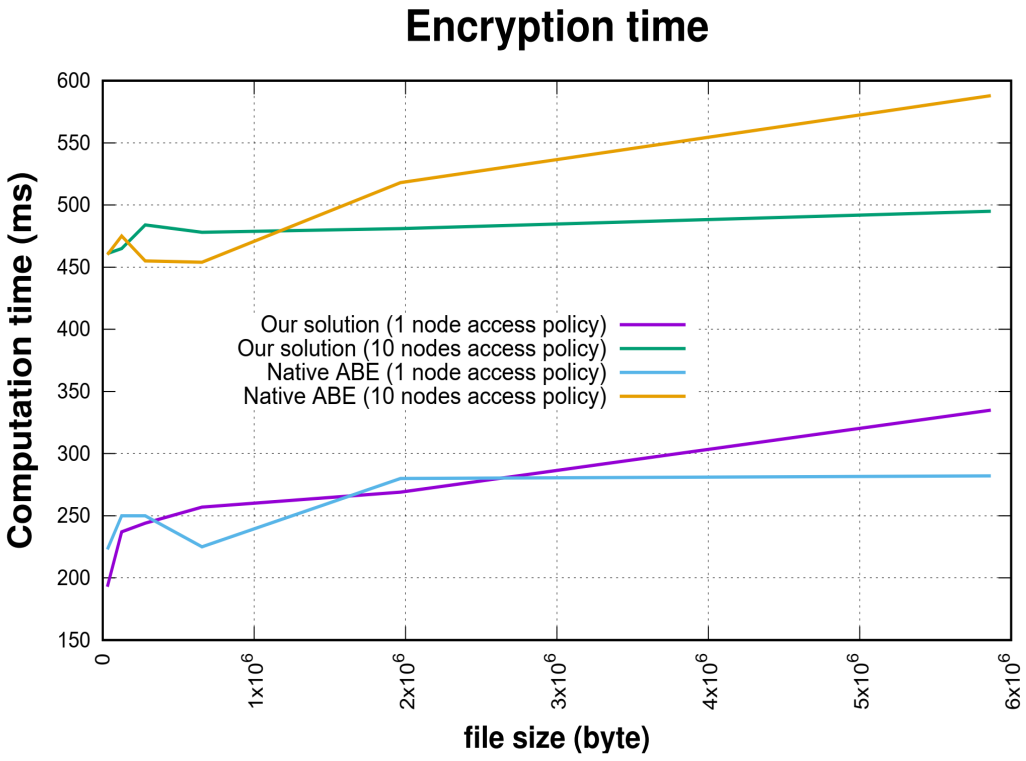



Figure 4.11 – Encryption time

As shown in figure 4.11, we can clearly see that the encryption time is not affected by the size of the encrypted files, and sometimes the encryption of a bigger file is quicker than a smaller one. These results are due to two main reasons:

1. In our implementation, we do not apply the attribute-based encryption directly on the original file. Instead, we generate a random key, then we use it as a symmetric key to encrypt the report file. Next, we use attribute-based encryption to encrypt generated symmetric key. Finally, we join both ciphertexts into a final one. Therefore, ABE scheme is actually applied on symmetric keys that have the same size, while the encryption of the data, which might endure a significative computational time, is replaced by a symmetric encryption which is very fast even with different data file sizes.

2. We recall that ABE ciphertext is defined as follows:

$$C_0 = Me(g_1, g_1)^s,$$
$$C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, g_1)^{\alpha_{\rho(x)} r_x},$$
$$C_{2,x} = g_1^{r_x}, C_{3,x} = g_1^{\beta_{\rho(x)} r_x} g_1^{\omega_x} \forall x$$

Where, several coefficients such as $s$, $\lambda$ and $\omega$, are randomly chosen from $Z_p$, and the algorithm uses them to compute exponentiations. Therefore, the computational cost depends of the chosen random values. Thus, given two different encryption processes, if the chosen random values during the first process are smaller compared to those chosen in the second one, it will result a small encryption time for the first process. This also explains the fact that sometimes the encryption of bigger files can be quicker than smaller ones.

Finally, we can clearly notice that the encryption cost is more or less the same between our solution and the native ABE. This means that the extra symmetric re-encryption that we have applied on some parts of the ciphertext, does not affect the performance of the original attribute-based scheme.

#### 4.6.0.3 Data decryption cost

As the encryption, we evaluated the decryption process of our access control solution and the native attribute-based decryption algorithm. In figure 4.12, we presents the average decryption time of several encrypted report files. We note that these results are issued from the decryption of different reports with several access policies.

Just as the encryption, we notice that our additional symmetric decryption on attribute-based scheme does not affect the performance of the original ABE, because the decryption time of the native ABE is approximately the same as ours.

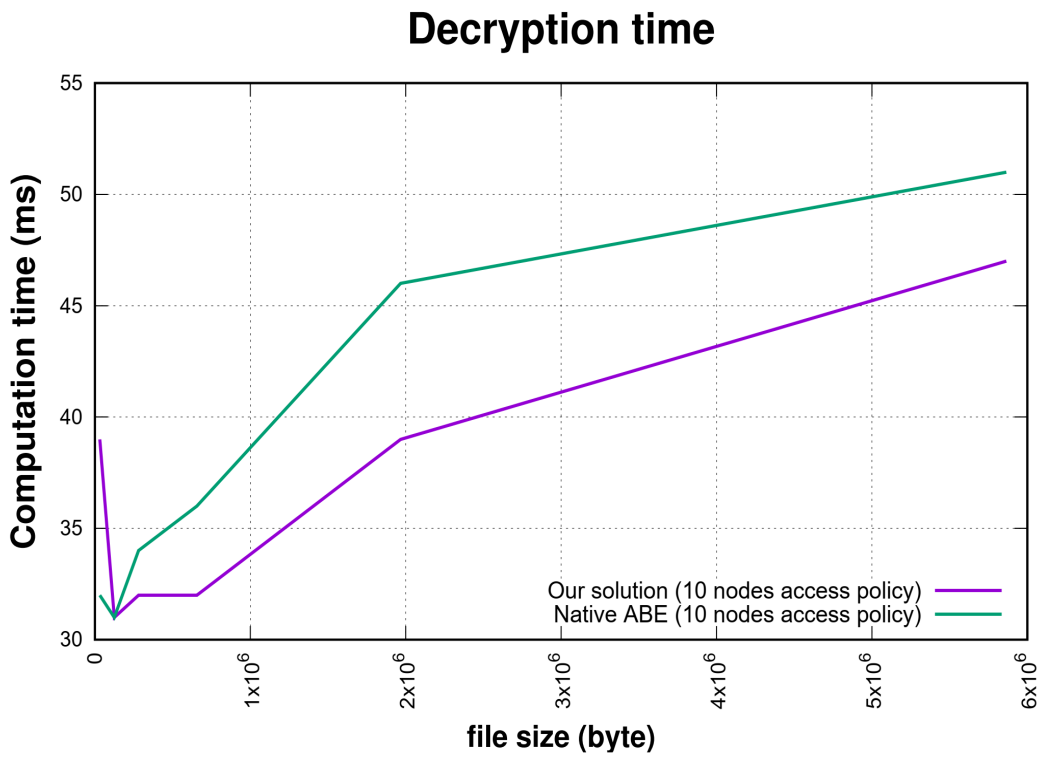



Figure 4.12 – Decryption time

#### 4.6.0.4 Revocation cost

We present in table 4.2 a comparaison of our solution and existing time-based solutions in terms of computation operations performed in the revocation. We note that we have not provided a comparison with proxy-based solutions since these proposals use a different architecture than ours. Furthermore, we provide in table 4.3 a comparison between our revocation solution and Yang et al.'s time-based solution [Yang et al., 2012]. We assume in our evaluation scenario that the revocation occurs each time that there is a team substitution, i.e. each eight hours. We recall that the time-based solution suggests to divide the time-space into slots and performs a key generation each time slot (eight hours in our scenario). On the other hand, in our solution we do not generate new attribute keys. Instead, we change only the secret of the attribute group, we compute a new complement for each non-revoked user, and finally we upload the updated complements on the cloud server. Note that we include the upload time in the measurement of our revocation scheme.

Despite the fact that we include the time of uploading complementary shares into the cloud, the evaluation results show that our solution presents better results compared to the time-based revocation. These results are logical since the time-based solution performs a new key generation each time slot, which leads the authority to compute exponentiations and thus it costs a significant computational time.

Contrariwise, our solution does not operate on the attribute key level. It changes only the revoked attribute secret and performs an update on non-revoked users' complements, which leads the authority to compute only division operations instead of exponentiations. Note that for a lower number of employees, such as the values chosen in Table 4.3, the time consumed in our revocation is due to the upload process considering that the computation of 50 division operations is negligible.

## 4.7 Conclusion

In this chapter, we have proposed a new attribute-based access control framework with an efficient revocation method for multi-authority architectures. Our solution ensures security requirements such as confidentiality, forward and backward secrecy and collusion resistance. In addition, we applied our solution on connected vehicle use case and proved its performance in term of encryption, decryption and revocation through experimentation. Our framework provides a secure, flexible and fine-grained access control, and deals efficiently with the revocation problem known in attribute-based systems, without launching any key regeneration process and performing any changes on the users' side. Furthermore, the authorities are not the only entities responsible for the revocation in our scheme. Henceforth, even the data owner can prevent other entities from getting access to its personal domain without relying on any third party. We go farther in our solution and provide a new efficient revocation mode for the fully distributed data-sharing model. In the future work, we intend to study the possibility of introducing proxy servers in the architecture to lighten the encryption and decryption cost on the entities with limited resources, while maintaining the same security level.

# *Mutual-authentication in fog computing architecture*

## 5.1 Introduction

Fog computing is a new paradigm which extends cloud computing services into the edge of the network. Indeed, it aims to pool edge resources in order to deal with cloud shortcomings such as latency problems. However, this proposal does not ensure the honesty and the good behavior of edge devices. Thus, security places itself as an important challenge in front of this new proposal.

Authentication is the entry point of any security system, which makes it an important security service. Traditional authentication schemes endure latency issues and some of them do not satisfy fog-computing requirements such as mutual authentication between end devices and fog servers. Thus, new authentication protocols suitable for this environment are needed.

In this chapter, we propose a novel, efficient authentication protocol which ensures mutual authentication at the edge of the network. Our scheme performs a first registration in the cloud level, and then it uses credentials provided by the cloud to realize any eventual mutual authentication between the users and the fog nodes, without any resort to the cloud. We base our construction on blockchain technology and secret sharing technique. The Blockchain is maintained by fog nodes and it allows end users to authenticate any fog node in the architecture. In addition, it allows fog nodes to establish mutual authentication with each other. It is true that blockchain is a resource consuming technology, but fog nodes fit its requirements at least in terms of storage. Moreover, unlike known public blockchains such as Bitcoin [Nakamoto, 2008], our blockchain is private and does not suffer from the substantial amount of computational power imposed by the proof of work in order to ensure synchronisation. In addition, some heavy tasks such as block validation, is dedicated to the cloud brokers (permissioned blockchain model). On the other side, end users are authenticated through secret sharing mechanism without using the blockchain. Indeed, since the number of end users is way too big compared to fog nodes, it

will not be efficient, in terms of storage, to register a huge number of users into a blockchain. Whereas, using our secret sharing scheme, fog nodes store only a few and a fixed number of information in order to authenticate any end user in the architecture.

Our solution is secure and provides the following advantages:

- Dynamic and scalable in terms of users joining the system, without any need for the brokers to contact each fog node when a user joins the system.
- Secure and fully distributed authentication mechanism with multi-cloud provider architecture.
- Low latency since both users and fog nodes perform authentication at the edge of the network without resorting to the cloud.
- Adaptive and portable scheme, which relies on the authentication systems that are already set up to realize a first identification of the users. Indeed, our solution does not require the creation of any new public key infrastructure (PKI), instead it builds its security basis on the already implemented security schemes in the cloud level and extends them to the edge of the network.
- Low overhead in terms of authentication.

The remaining of the chapter is organized as follows. In section 6.2, we give backgrounds on Shamir's secret sharing scheme and blockchain technology. After that, we present our solution in section 6.3. In section 6.4, we present the threat model. Then, we present our security analysis in section 6.5. We evaluate the performance of our solution and its complexity in section 6.6. Finally, we conclude in section 6.7.

## 5.2 Background

In this section, we present some security models that will be used in our authentication solution.

### 5.2.1 Review on Shamir's secret sharing scheme

In cryptography, secret sharing refers to a method for distributing a secret amongst a group of participants by giving each one of them a part of that secret. These parts are called shares. The distributed secret can be reconstructed if all the shares are combined together.

Based on the fact that the collection of at least $k$ different points can reconstruct a polynomial of degree $k-1$, Shamir [Shamir, 1979] introduced the secret sharing scheme by dividing a secret $S$ into pieces $(x_i, S_i = q(x_i))$ using a randomly chosen polynomial:

$$q(x) = a_0 + a_1 x + a_2 x^2 + ..... + a_{k-1} x^{k-1}$$

In which $a_0$ represents the secret S and $(1, S1 = q(1)), (2, S2 = q(2)) \ldots (k, Sk = q(k))$ are the shares. The polynomial q can be reconstructed using Lagrange interpolation as:

$$q(x) = \sum_{i=1}^{k} Y_i \times \prod_{j=1, j \neq i}^{k} \frac{x - x_j}{x_i - x_j}$$

Where $Y_i = S_i$

Consequently, the secret $S$ can be calculated as $S = q(0)$

### 5.2.2 Review on Block-chain

Blockchain is a new promising technology that revolutionized the world of cryptocurrency these last years. The main aim of this technology is to allow heterogeneous nodes to communicate and exchange assets between them. This exhange is done in a completely distributed and secure way, without relying to any trusted central entity. Basically, each node in the blockchain does not trust any other node, but, it trusts the whole blockchain network. In the blockchain, each node holds a pair of cryptographic keys (public and private key) that allows to generate transactions and interact with other nodes in the network. In addition, these transactions are immutable. Indeed, it is hard to falsify any transaction once added to the blockchain. In the distributed blockchain network architecture, it is mandatory that the whole nodes reach a consensus state in the validation of each transaction. The validation process consists generally in solving a heavy computational problem. This mechanism endows the blockchain with the immunity property. Indeed, to falsify or update one block already validated, an attacker needs to realize the same heavy validation process for this block and all its subsequent blocks in the blockchain. In what follows, we explain the different steps from transaction generation until the validation of the transaction in the blockchain [Christidis and Devetsikiotis, 2016]:

- Fist, when a node wants to exchange some assets with another node, it generates a transaction containing the asset and signs that transaction with its private key. Then, it broadcasts the transaction to all the peers in the blockchain.

- Next, the miner nodes periodically gather a set of transactions in one single

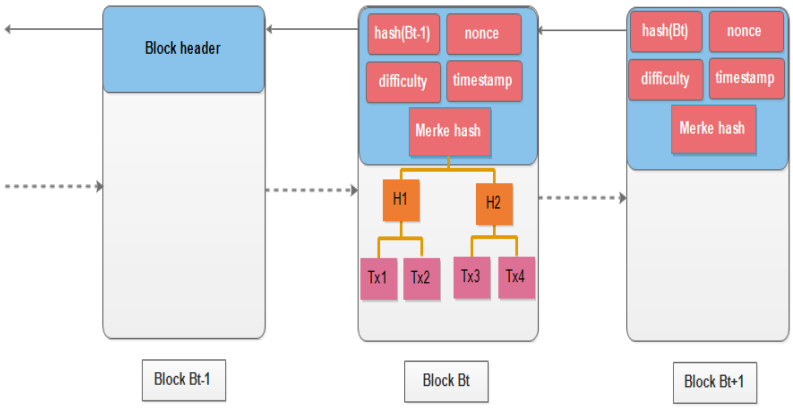



Figure 5.1 – blockchain structure

block, and proceed to the validation process which consists on solving a hard mathematical problem.

- Finally, the other nodes verify the format of the transaction and the correctness of the solution, that the miner proposed for the hard mathematical problem. If the format of transactions and the proposed solution are correct, then, the nodes add this block to the blockchain. Otherwise, the block is discarded. Figure 5.1, describes the general structure of a blockchain.

## 5.3 Our solution

In this section, we present our proposed solution which ensures mutual authentication in fog computing architecture. The notations that we will use in the presentation of our solution are given in Table 5.1.

In our solution, we consider an architecture (figure 5.2) composed of the following components:

- Several cloud brokers responsible for the verification of users and fog nodes' identities. In addition, these entities distribute authentication credentials for both users and fog nodes in order to allow them to be authenticated at the edge of the network. We assume that each cloud broker $B_i$ already has a pair of keys (public key $PK_i$ and private key $SK_i$). The key $PK_i$ should be known by the other brokers since the broker $B_i$ uses $SK_i$ to sign each transaction that it generates. In addition, the brokers should share in common another pair of keys $(PK, SK)$. $SK$ will be used to sign valid transactions, while $PK$ will be used by the users to verify the $SK$ signature at the edge authentication level.

| Notation | Description |
|---|---|
| $H(*)$ | Hash Function |
| $P(x)$ | Polynomial of degree $m$ |
| $(PK_i, SK_i)$ | Broker $B_i$'s public and private keys respectively |
| $(PK, SK)$ | Validation public and private keys respectvely, shared between all the brokers |
| $n$ | a public parameter which defines the group $Z_n$ |
| $(X_{ui}, Y_{ui})$ | User $ui$'s coordinates generated by one of the brokers |
| $X_{ui}^{-1}$ | private key related to the public key $X_{ui}$ |
| $F_{si}$ | Fog node $i$'s share |
| $(FPK_i, FSK_i)$ | Fog node $i$'s public and private keys |
| $B_i$ | Broker $i$ |
| $CS$ | Session key |
| $H[]$ | The Hash chain |
| $\sigma$ | Cryptographic digital signature |
| $\{M\}_K$ | The encryption of the message $M$ with the public key $K$ |

Tableau 5.1 – Table of notations

- Fog nodes, which provide computational services at the edge of the network.
- End devices (users), which request services from the fog nodes.

### 5.3.1 The main idea of our solution

In our solution, an application called the broker is set up in each cloud in order to verify the authenticity of both the users and the fog nodes. Each user must perform a first authentication with one of the cloud brokers that consequently verifies the validity of the user's identity. If this verification succeeds, the cloud broker will generate credentials and sends them to that user. Providing this credentials to the user will allow any fog node to authenticate him at the edge of the network. To authenticate a user, the fog nodes perform a first authentication using their certificates at the cloud as well. The aim of this step is to verify the authenticity of the fog nodes and provide some information which allows them to verify users' credentials at the edge of the network without contacting the cloud brokers. In addition, the brokers set up a mechanism which allows the users to authenticate the fog nodes by using blockchain technology. Indeed, after the verification of fog node's certificate, the cloud broker generates a transaction which contains the node's public key and signs it using its private key. Then, it broadcasts that transaction so that one of the other brokers can validate and insert it into the blockchain. We note that

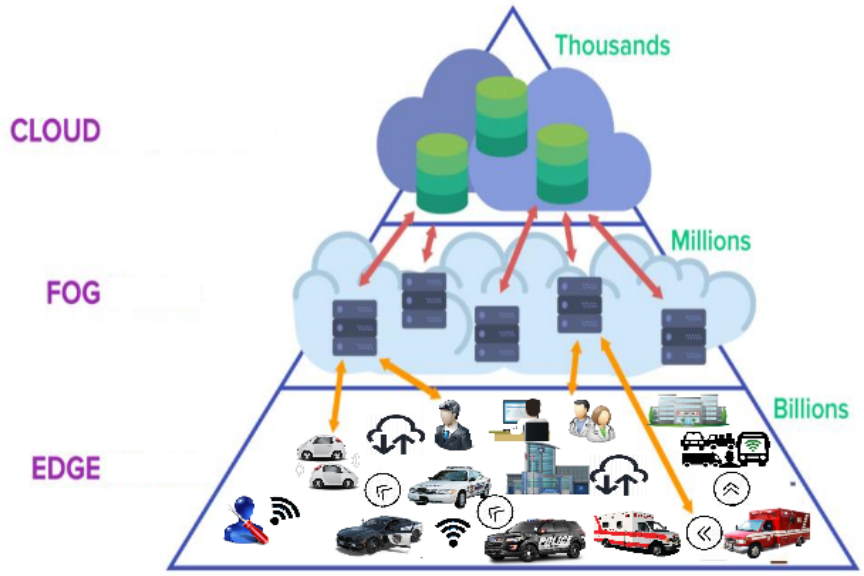



Figure 5.2 – Fog-computing architecture

our blockchain is private and it is stored in each fog node. Furthermore, it does not just allow the users to authenticate any fog node in the architecture, but it also allows fog nodes to authenticate each other.

The mutual authentication starts when a user requests a service from a fog node. First, that user should authenticate the fog node from which he requests a service. Thus, it verifies the part of the blockchain where one of the cloud brokers has signed the transaction that contains the public key of that fog node. Once the user verifies the authenticity of the fog node, he should send his credentials to that fog node. Then, based on secret sharing scheme, the fog node combines the user's credential and the information provided by the cloud broker, during its initial authentication, to verify the authenticity of that user.

We note that in our scheme, we do not consider further access control issues with respect to whether the user has the right to run any application in the fog node, or which services he has the right to exploit.

### 5.3.2 Implementation

In what follows, we show how we can achieve our proposed authentication scheme which allows to verify the authenticity of both the users and the fog nodes at the edge of the network.

We note that our solution uses public key cryptography in some points of the authentication process, thus, for sake of illustration in what follows, we consider RSA [Rivest et al., 1978] as a model of public key cryptography.

Our scheme achieves mutual authentication based on secret sharing scheme and blockchain technology, and it works as follows:

#### 5.3.2.1 Setup phase

In this phase, the brokers set up the system parameters that are going to be used in the eventual registration and authentication phases. We note that it is sufficient that only one of the brokers runs this setup phase and share the setup parameters with the other brokers. Thus, in what follows, let us consider that only one of the brokers is running this phase as:

- Initialize a blockchain that will contain the public key of each legitimate fog node.
- Choose a polynomial $P$ of degree $m$, as follows:

$$P(x) = a_0 + a_1 x + a_2 x^2 + ..... + a_n x^m$$

  The degree $m$ of the polynomial can be randomly chosen and does not depend neither on the number of users nor the number of fog nodes. $a_0$ is considered as the secret token that is going to allow the fog nodes to verify the authenticity of the users, and $a_i, i \in [1, m]$ are randomly chosen coefficients from $Z_p$.

- Choose two primes $q_1$, $q_2$ and compute two values $\phi(n) = (q_1 - 1) \times (q_2 - 1)$ and $n = q_1 \times q_2$.
- Generate $m$ points $Pi(X_i, Y_i)$ randomly from $Z_p$, and set the verification parameters $VP$ as:

$$VP = \{(token = S), \{P_i(X_i, Y_i)\}, n\} \tag{5.1}$$

#### 5.3.2.2 Fog registration phase

Fog nodes should perform a registration in the cloud level and provides its certificate to one of the cloud brokers. Then, the broker runs the following actions:

- Verify the certificate given by the fog node.
- Generate a transaction that is signed by the secret key $SK_i$. This transaction contains the following information: the public key of the fog node along with its current state ("*legitimate*" or "*malicious*" fog node). Note that in our solution, the state "*legitimate*" is the default state.

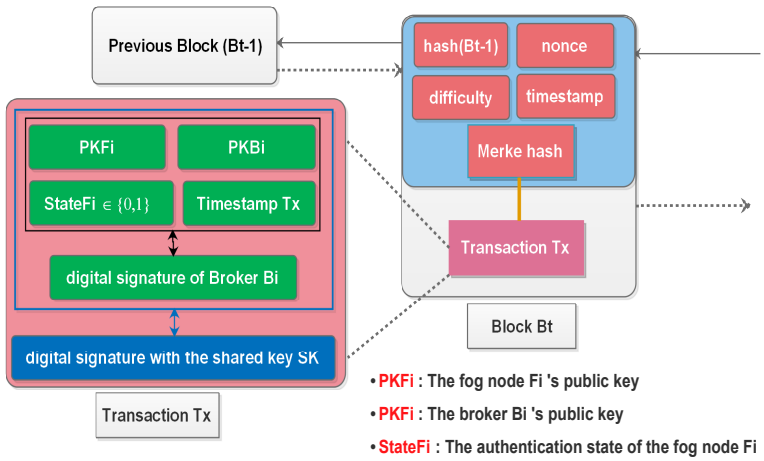



Figure 5.3 – structure of our blockchain

- Insert the transaction into a new block and fill the "difficulty" field (see figure 5.3) which defines the mathematical problem that should be solved in the validation step.

- Broadcast the transaction between the blockchain peers (the other brokers) in order to be validated.

To validate the transaction, the brokers run the proof of stack algorithm, which designates one of the brokers $B_j$ to verify and validate the transaction as follows:

- Verify the signature of the transaction using the broker's $B_i$ public key $PK_i$

- If the signature has been successfully verified, solve the mathematical problem defined through the difficulty field in $B_i$'s block.

- Fill the solution of the mathematical problem in the nonce field, then sign the transaction using the validation key $S_k$

- Insert the new block into the blockchain.

We note that the verification in this step has no relation with the certificates' verification that the broker ran before; it only consists of verifying that one of the valid brokers has generated the transaction. Figure 5.3, shows the structure of our proposed blockchain.

Once the fog node's public key $FPK_i$ has been inserted into the blockchain, the broker $B_i$, who verified its certificate, sends to that legitimate fog node the verification parameters, computed in the setup phase, in order to allow to authenticate users without resorting to the cloud.

#### 5.3.2.3 Users registration phase

The users should also perform a first registration in the cloud level. To confirm its identity, a user needs to successfully be authenticated using the already adopted authentication system in the cloud. After that, the cloud broker generates new credentials to that user in order to allow him to perform any eventual authentications at the edge of the network (fog node level) as folllows:

- Choose a unique and random $X_{ui}$ from $Z_p$ which is coprime with $\phi(n)$. Then, it computes a $X_{ui}^{-1}$ which is the modular multiplicative inverse of $X_{ui}$ (modulo $\phi(n)$).
- Generate a unique point $P_{ui}(X_{ui}, Y_{ui})$, where $X_{ui}$ and $Y_{ui}$ in $Z_p$. We note that $P_{ui}$ has to be different from the $P_i$ points generated in the setup phase.
- Combine the user's specific point with the $m$ points $P_i$ generated in the setup phase as follows:

$$L_{u,i} = \prod_{j=1}^{m} \frac{X_{ui}}{X_{ui} - X_j}$$

$$Us_{ui} = Y_{u,i} \times L_{u,i}$$

- Prepare the user's credential as:

$$\text{User's credential} = \{PK, Us_{ui}, X_{ui}, X_{ui}^{-1}, n\} \tag{5.2}$$

  Where: $PK$ is the validation public key.

We note that the operations to compute the $L_{u,i}$ and the user's share $Us_{ui}$ are realized in $Z_p$ and do not have any relation with $Z_n$, where $n$ has been defined in the setup phase. In addition, by sending the public key ($PK$), the user is allowed to verify that the information given by the fog node, in the mutual authentication phase, comes from the valid blockchain and not a falsified one.

#### 5.3.2.4 Mutual authentication phase

Using the credentials given by the cloud broker and the information in the blockchain, both the users and the fog nodes can mutually authenticate each other at the edge of the network as follows:

**Fog node authentication:** the user starts by authenticating the fog node through the following steps:

- The user requests the transaction from the blockchain in which the cloud broker inserted and signed the fog node's public key and its current state.

- As soon as the fog node sends back its transaction block, the user verifies that the received transaction comes from the valid blockchain that the brokers use to publish legitimate fog nodes. Therefore, given a transaction block defined as:

$$Bc_i = (header, Tx, H(Tx)\sigma_{SK})$$
$$Tx = (Fn_i, H(Fn_i)\sigma_{SK_i})$$
$$Fn_i = (FPK_i, state, timestamp)$$

  Where:

  - ◇ *header* the block $Bc_i$ header in the blockchain
  - ◇ *state*= valid or not valid.

  The user computes:

$$H_1 = H(T_x)$$

  Where: $H$ is a hash function

- Verify the signature of the block using the validation public key $PK$, received as part of its credentials as follows:

$$H_2 = (H(T_x)\sigma_{SK_i})^{PK}.$$

- If $H_1$ is equal to $H_2$ then the fog node trasaction is verified. Otherwise, the user notices that the fog node did not provide a block from the valid blockchain since the signature does not match with the public key $PK$ provided by the broker.

**User authentication:** once the user verifies the transaction presented by the fog node, it starts its authentication process. Therefore, it sends its credentials encrypted with the fog node's public key as:

$$Credentials = \{Us, X_{ui}\}FPK_i$$

Where:

$$Us = L_{u,i} \times Y_{u,i}$$

On the other side, the fog node verifies the user's authenticity as follows:

- Decrypt the received verification parameters using the private key $FSK_i$.
- Perform a polynomial interpolation in $Z_p$ using the values $(Us_{ui}, X_{ui})$ provided by the user, and the $(X_i, Y_i)$ coordinates provided by the cloud broker (eq.1) as follows:

$$Lfn_k = \frac{-X_{ui}}{X_k - X_{ui}} \times \prod_{j=1, j\neq k}^{m} \frac{-X_j}{X_k - X_j}$$

$$Fs = \sum_{i=1}^{m} Y_i \times Lfn_i$$

$$\text{Computed token} = Us_{ui} + Fs = S'$$

- Compare the two values $S'$ and the token $S$. If the token $S'$ is valid, the fog node generates a hash chain $H[]$ and a session key $CS$, which will be used in the eventual further data exchange between the fog node and the authentic user. Then, it encrypts them using $X_{ui}$ as follows:

$$\text{access credential} = \{CS, H[]\}X_{ui}$$

  Where:

  - $\{*\}X_{ui}$ is a public encryption method in $Z_n$, which uses $X_{ui}$ as a public key.

  As we can notice, we linked both secret sharing scheme and public key encyption through $X_{ui}$ value. Thus, this value is not just an importatnt part in the process which prooves that the user is valid member of the group, it also respresents an insurance that the access credentials given by the fog node can only be decrypted an entity which really possesses credentials (eq.5.2) given by the broker.

- Finally, the fog node sends the access credentials, and triggers a timer which defines the period of time that the fog node should wait until the first service request from the user.

We note that, if the user does not send any service request, encypted with the session key and contains the first element of the hash chain $H[0]$. Then, by the end of the timer, the authentication session expires. Figure 5.4, describes the mutual authentication process.

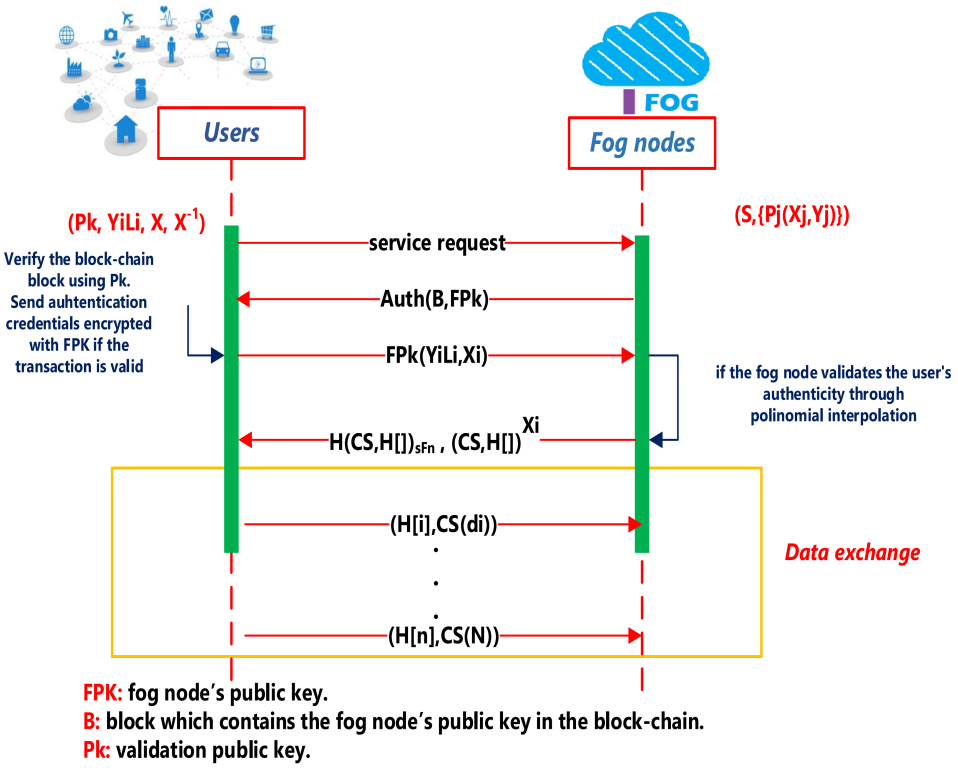



Figure 5.4 – edge network mutual authentication

## 5.4 Threat model

In our protocol, we distinguish two different adversarial models where each model reflects a specific situation defined as follows:

1. **Case 1: an attacker impersonates fog nodes:** let $A$ be a polynomial time adversary which interacts with a signature oracle. Thus, it submits arbitrary messages $m_i$ to the oracle in order to get the signature of these messages. Finally, the adversary outputs a message $m$ that has never been submitted to the oracle along with its signature.

   Adversary $A$ wins the security game if he outputs a valid signature for the message $m$.

2. **Case 2: an attacker impersonates end-users:** let $A$ be a polynomial time adversary which interacts with a random encryption oracle. $A$ submits two messages $\{m_0, m_1\}$ to the oracle. Then, the oracle picks a random coin $b \in \{0, 1\}$ and replies by sending $E(m_b)$, where $E$ is a public key encryption function. Finally, the adversary outputs a guess $b'$ about which one of the two submitted messages $\{m_0, m_1\}$ has been encrypted with $E$.

The advantage of adversary $A$ in this game is expressed as:

$$Adv = Pr[b = b'] - 1/2$$

## 5.5 Security analysis

In this section, we prove that our authentication solution ensures the expected security requirements.

### 5.5.1 Replay/impersonation attack

In our authentication scheme, the fog node verifies the user's credentials, then it sends him the session key with the hash chain encrypted using $X_{ui}$ as a public key. Finally, it sets a timer and waits for the user's request. On the other side, the user needs to get the session key and sends a service request to the fog node before the achievement of the timer. Otherwise, the authentication session will expire. As we can notice, the user needs to send a service request in a limited period of time. Thus, it will be useless for any party to try to replay the user's authentication request since any party, which wants to successfully perform this attack, needs to recover the user's private key $X_{ui}^{-1}$ and get the session key to use it in the eventual service request. Since $X_{ui}^{-1}$ is a secret key generated through one of proven secure public key schemes, as RSA [Rivest et al., 1978], its security is preserved. Likewise, it remains useless to impersonate the user's identity and use its credentials to be authenticated in the fog node, since it also requires the attacker to recover the user's private key $X_{ui}^{-1}$.

On the other side, if an attacker impersonates an existing fog node identity, it will need to recover the private key of that fog node, which is used to sign access credentials (the session key and the hash chain). Likewise, if an attacker tries to convince a user that he is a legitimate fog node, it needs to provide a valid blockchain transaction signed by one of the known brokers and which contains its public key. Therefore, the attacker has to forge the validation signature key used by the brokers. In the case of RSA signature, a formal proof about its security has been provided in [Cramer and Shoup, 2000].

### 5.5.2 Man in the middle

If an intermediate node tries to perform man in the middle attack to get access in one of the legitimate fog node, it will need to guess the user's private key $X_{ui}^{-1}$, in order to find out the session key and send back a service request to the fog node before

the achievement of the timer. Thus, this attack will also fail since the probability of guessing the user's private key in a limited time is negligeable.

### 5.5.3 User/ Fog compromise

If a fog node has been compromised, it will not affect the authentication of the users nearby other fog nodes since fog nodes possess only verification parameters and have no knowledge about the users' private keys $X_{ui}^{-1}$. Thus, a compromised fog node cannot perform any kind of attack which aims to use any user's credential to get access in other fog nodes. On the other side, a user which has been compromised, can still be authenticated in any other fog node. Thus, it is important that the users ask for a revocation nearby one of the cloud brokers in case they were compromised. If the system detects misbehavior in any user/fog node, one of the cloud brokers needs to revoke them. Using the blockchain as a repository of the revocation list, for both revoked fog nodes/users, could be an adequate solution to manage this situation.

## 5.6 Performance evaluation

In this section, we evaluate the performance of our authentication scheme. Our experimentations have been realized in a real wireless adhoc network, using two laptops (an HP, i7 laptop with a CPU frequency of 2.7 GHZ and a Samsung i5 laptop with a CPU frequency of 2.6 GHZ). We first measure the computational time that the broker spends in the generation of users' credentials during the users' registration phase. Then, we provide the measurements of our edge authentication level and compare it with multi-level certificate-based solution. We note that all arithmetic operations are realized in $Z_n$ or $Z_p$ where $p$ and $n$ are encoded in 1024 bits (128 bytes).

### 5.6.1 Registration in the Cloud

The registration algorithm in the cloud broker level verifies the user's identity, then it generates credentials for that user. The verification of users' identities depends on the authentication algorithm adopted in the cloud. Whereas, the credential generation step consists only of computing some multiplications. Thus, we conclude that the complexity of this phase is linear in the order of $O(n)$, where $n$ is the number of registration requests that the broker receives at the same time. Figure 5.5, illustrates the computational time of credential generation according to the number of registration requests, received in parallel.

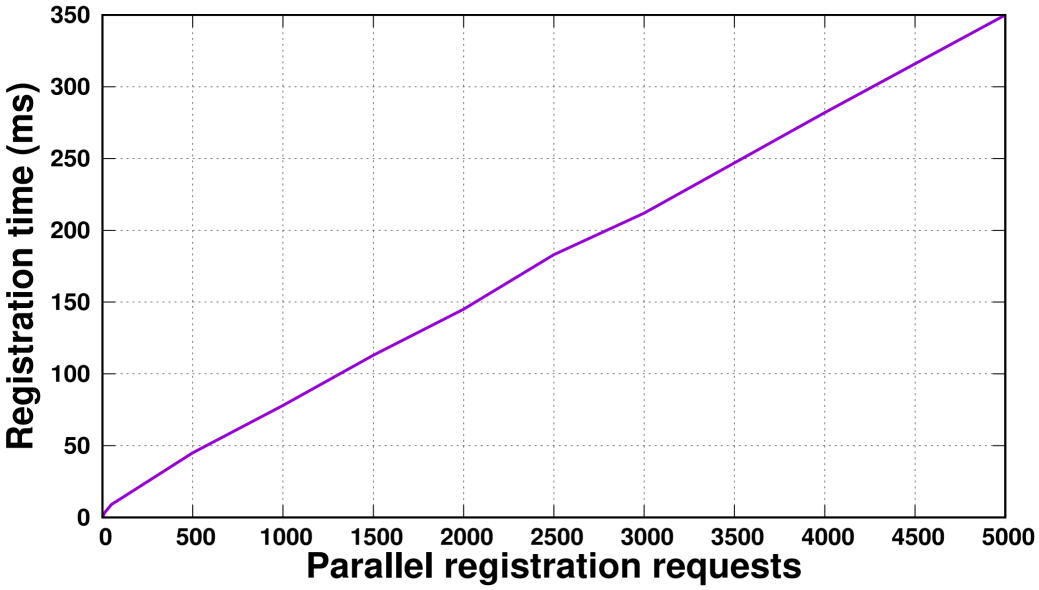



Figure 5.5 – Registration phase

Tableau 5.2 – Transactions' verification and validation time

| | High | Low | Average |
|---|---|---|---|
| Transaction validation time(ms) | **29000** | **3000** | **10487** |
| Transaction verification time(ms) | **0.0481** | **0.0211** | **0.0482** |

### 5.6.2 Edge level authentication

As shown in Figure 5.6, the edge level authentication does not take much time. At this authentication level the fog node verifies the user's credential. In our solution, almost all computation operations are performed by the fog node, which has a considerable computation power. In addition, the authentication process is performed at the edge level of the network so it does not occur a considerable latency. In our solution, the fog node will only perform a constant number of multiplication and addition operations to verify the authenticity of the user. Similarly, to authenticate a fog node, the user will only verify the transaction provided by that node. This operation consists of verifying that one of the authorized brokers signed the provided transaction, which is not a time consuming task as shown in Table 6.2.

In existing certificate-based solutions, the fog node will search and verify a set of intermediate certificates going up to a certificate issued from one of the root

authorities that the fog node trusts. This operation endures an important latency since the verification depends on the number of intermediate authorities going up to a root authority (certificate level). Note that in our experiments, the intermediate authorities are in the same network. Thus, the latency can be higher if the authorities are on another network.

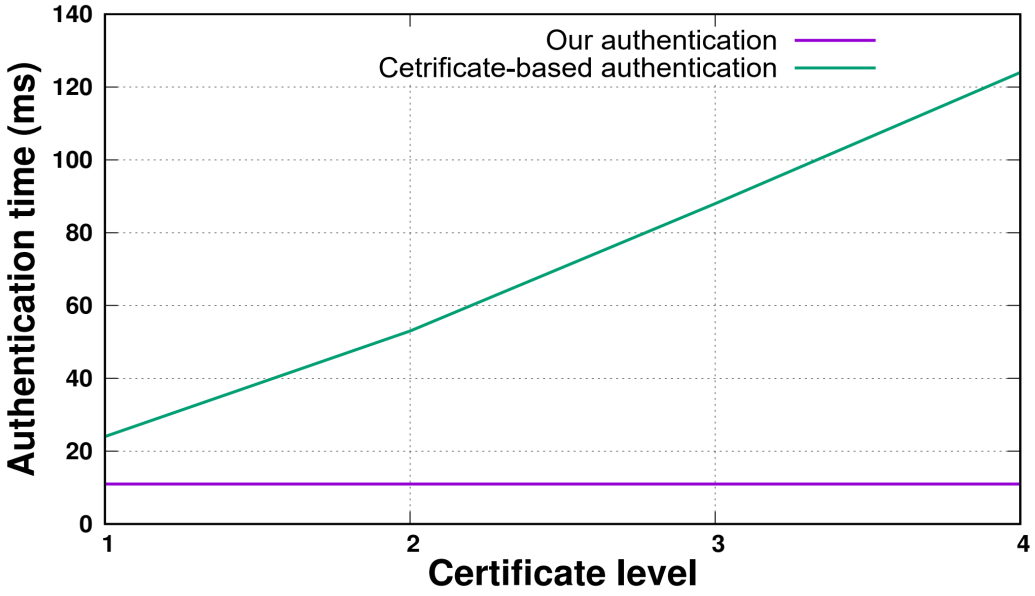



Figure 5.6 – Our solution Vs certificate-based authentication

Table 6.1 shows the computation and storage overhead in each step of our protocol. As we can see, all the components perform lightweight arithmetic operations during the different steps of our protocol, except Cloud brokers which sometimes validate blocs in the blockchain. In terms of storage, the users store few credentials while the fog nodes store the blockchain and verification parameters. Note that the size of the blockchain depends on the number of fog nodes.

### 5.6.3 Blockchain Performance evaluation

In order to evaluate the performance of validation and verification of transactions, which are part in the process of fog nodes registration, we have measured the average time to validate one transaction as well as the time that the user takes to verify the signature and the content of one transaction. In our evaluation, we use go-ethereum platform [Git, 2018], which is one of the official implementations of ethereum blockchain protocol. In table 6.2, we presents the average time of transaction's validation and signature verification. In this test, we also measure the average memory and CPU occupations. We note that the average memory usage for running the mining process is around 27.9 MO. During the transactions' validation,

Tableau 5.3 – Storage and computation cost in our scheme

| | Storage (bytes) | Computation | | |
|---|---|---|---|---|
| | | user registration | fog registration | mutual authentication |
| Cloud broker | / | 1 Inv + $2m$ Mult + $2m$ Add | 1 Sign + Val | / |
| Fog node | $nbF \times TS + (2m+3) \times 128$ | / | / | 1 Asm-Dec + $2m$ Add + $2m$ Mult |
| End user | $5 \times 128$ | / | / | 1 Sign-Verif +1 Asm-Enc |

(Inv, Mult, Add) refer to modular inverse, multiplication and addition resp. (Sign, Sign-Verif, Asm-Enc/Dec) refer to RSA signature, signature verification and RSA asymmetric encryption/decryption resp. $(m, nbF, TS)$ are the predefined polynomial degree, number of fog nodes and trasanction size resp. Val refers to the validation process in the blockchain

the percentage of miner's CPU overhead reaches 92.65%. We note that the mining operation is done by the cloud brokers, which have an important computation power far away from what we use to evaluate our scheme's performance. Therefore, better results can be achieved as much as we use more computational power.

## 5.7 Conclusion

In this chapter, we have proposed a new secure authentication scheme based on secret sharing and blockchain technology in fog computing architecture. In our scheme, both the users and the fog nodes perform one registration in the cloud level. Then, they will be able to mutually authenticate each other at the edge of the network without resorting to the cloud. The users hold some information which allow them to verify the authenticity of any legitimate fog. Moreover, fog nodes in our solution do not need to store any users' identifiers and any digital certificates. They only hold a couple of values that are going to allow them to verify the authenticity of any user in the system. In addition, fog nodes can also authenticate each other at the edge of the network using the blockchain. This feature is essential especially in the context of secure VM migration from a fog node to another. Furthermore, our scheme deals efficiently with situations where an entity from the system tries to impersonate another one in order to get services from fog nodes. Finally, our experimental results show that our proposal realizes mutual authentication in a short time since most operations are performed at the fog nodes level. In the future, we intend to address

more intensively the revocation problem in fog computing architecture.

# An Efficient Accountable Privacy-Preserving Scheme for Public Information Sharing systems

## 6.1 Introduction

Since the emergence of data externalization technologies, as cloud or fog computing, privacy has become a major concern for all users. In fact, service providers might use users' personal information for other purposes such as behavior detection, preference detection, activity tracking, etc. It is true that most of the providers use this information to provide a comfortable service or to gain benefits by selling information to advertising companies, but it still violates users' privacy since it is usually done without users complete approval [Keshavarz and Anwar, 2018]. Several scientific research papers in the literature have proposed to deal with privacy issues using existing anonymization techniques [Ji et al., 2016], but few of them considered traceability service. Whereas, when security systems do not adopt traceability mechanisms, full anonymity may encourage users to act maliciously. In this chapter, we propose a novel privacy preserving solution with traceability service for public information sharing applications.

In our solution, communicating entities perform a first registration with a trusted authority, which provides access credentials to each registered entity. These credentials allow the authority to trace any entity in the network. We note that in our solution the trusted authority is the only entity that is able to trace other communicating entities.

To allow traceability in this information-sharing model, each information needs to be signed by its owner. Moreover, a specific application deployed in the fog servers will be in change of the the verification of the signatures to check out that the trusted authority can trace the origin of the information.

In order to fulfill this requirement while preserving the privacy of communicating entities, we propose to randomize the signatures provided with public information.

Randomizing the signatures is an efficient manner to preserve entities' privacy. Indeed, if an entity submits a new random signature at each information-sharing event, fog severs will not be able to trace the origin of the information. Nevertheless, since we also need to ensure traceability service in our information-sharing model, we propose to randomize the credentials provided during the registration phase. Randomizing these credentials will keep the privacy-preserving feature, but it also allows the application installed in the fog server to find out whether the authority could trace the shared information or not, without violating the information owner's privacy.

Our solution provides the following advantages:

- Anonymous information sharing model for any communicating entity in the network.
- Traceability of any communicating entity in the network.
- Completely decentralized information sharing model that does not require any interaction with the trusted authority during the information sharing process.
- Low communication and storage overhead for both fog servers and communicating entities.

The remaining of this chapter is organized as follows. In section 6.2, we give backgrounds on Schnorr signature scheme and bilinear maps. After that, we present our solution in section 6.3. In section 6.4, we present the threat model. Then, we present our security analysis in section 6.5. We evaluate the performance of our solution and its complexity in section 6.6. Finally, we conclude in section 6.7.

## 6.2 Background

In this section, we present some mathematical notions and security models that we are going to use in our traceable privacy-preserving scheme.

### 6.2.1 Bilinear Maps

Let $G_0$ and $G_1$ be two multiplicative cyclic groups of prime order $p$. Let $g$ be a generator of $G_0$ and $e$ be a bilinear map,

$$e : G_0 \times G_0 \rightarrow G_1$$

The bilinear map $e$ has the following properties:
Bilinearity: for all $u, v \in G_0$ and $a, b \in Z_p$ , we have:

$$e(u^a, v^b) = e(u, v)^{ab}$$

Non-degeneracy: $e(g, g) \neq 1$

We say that $G_0$ is a bilinear group if the group operation in $G_0$ and the bilinear map $e : G_0 \times G_0 \rightarrow G_1$ are both efficiently computable.

Notice that the map $e$ is symmetric since

$$e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$$

### 6.2.2 Schnorr signature scheme

Let $G$ be a group of prime order $q$, with generator $g$ and in which the discrete logarithm problem is assumed to be hard. Le $Z_q^*$ be a multiplicative finite field of prime order $q$. $H()$ denotes a collision resistant hash function. Assume that a signer $S$ has a private key $x$ and the corresponding public key $y = g^x$. To sign a message $m$, $S$ chooses a random number$k \in Z_q$ and computes $r = g^k, s = k - x.H(m, r)$. Then, the tuple $(m, r, s)$ becomes a valid signed message. The validity of signature is verified by $g^s.y^e = h(m, r)$. Schnorr signature [Schnorr, 1989] has been proven to be secure under the random oracle model in [Pointcheval and Stern, 1996]; where the authors have shown that existential forgery under the adaptive chosen message attack is equivalent to the discrete logarithm problem.

## 6.3 Our solution

In this section, we present our proposed solution which ensures traceable privacy-preserving information sharing in Edge-computing architecture.

In our solution, we consider an architecture composed of the following components:

- **Communicating entities** such as connected vehicles, connected objects or any entity interested in sharing public information in the network.
- **Externalization servers** such as Edge servers or Cloud which are responsible for information sharing. These components are semi-trusted and thus we assumed that they correctly perform the required tasks, but they may try to violate communicating entities' privacy.
- **Trusted authority** which is responsible for the registration of communicating entities besides ensuring traceability service in the network.

Figure 6.1 illustrates our considered architecture.

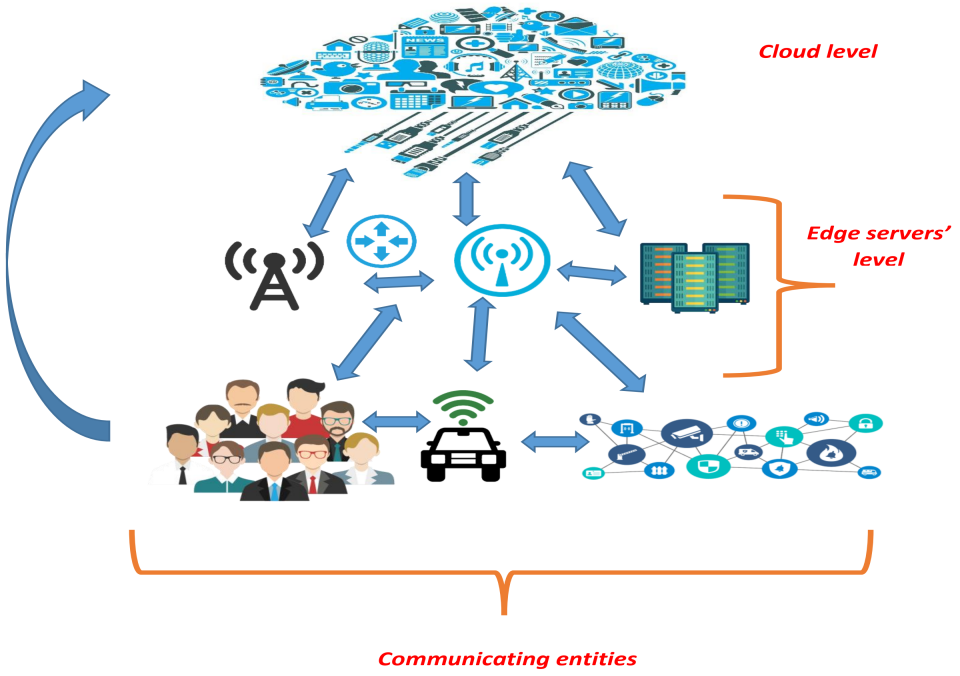



Figure 6.1 – A descriptive diagram of our architecture

### 6.3.1 Our construction basis

Privacy preserving and traceability features in our solution are two sides of the same coin. Indeed, in one hand, the users could share their data in a completely anonymous manner without being known neither by the fog servers, nor any other regular member within the information-sharing group. Therefore, users cannot be tracked in their eventual information-sharing activities.

On the other hand, and despite the fact that users' signatures are anonymous, our solution allows fog servers to find out whether the user is allowed to share information in the sharing group or not. Moreover, if the system detects any anomaly in the sharing group, our solution ensures that the trusted authority will trace the origin of any shared information.

Our construction is based on the following idea:

Given two polynomials $P_1$ and $P_2$ defined as follows:

$$P_1(x) = R_1 x + S$$

$$P_2(x) = R_2 x + S$$

Where $R_1, R_2$ are random values in $Z_p$ and $S \in Z_p$ is a common random value used in both polynomials. If we consider two random values $x_1$ and $x_2 \in Z_p$, the points $P_1(x_1), P_2(x_1)$ and $P_1(x_2), P_2(x_2)$ will result different random values. However,

as it is shown in equations 6.1 and 6.2, even if we use two different polynomials to generate points ($P_1(x_1), P_2(x_1)$ for example), computing the polynomial interpolation at $x = 0$ will always result the same value $S$, given the same values $(x_1, x_2)$. Therefore, we conclude that even with two different polynomials as defined bellow, we can always find the same secret $S$ if we use the same $x_i$ values to generate points, and then we compute polynomial interpolation at $x = 0$.

$$P_1(0) = \sum_{i=1}^{2} P_1(x_i) \times L_i = \sum_{i=1}^{2} P_1(x_i) \times \prod_{j=1}^{2, j \neq i} \frac{-x_j}{x_i - x_j} = S \tag{6.1}$$

$$P_2(0) = \sum_{i=1}^{2} P_2(x_i) \times L_i = \sum_{i=1}^{2} P_2(x_i) \times \prod_{j=1}^{2, j \neq i} \frac{-x_j}{x_i - x_j} = S \tag{6.2}$$

In our solution, we provide the externalization servers with constant values (computed using $x_1$ and $x_2$). On the other hand, the users submit points generated using $x_1$ and $x_2$ but through a new random polynomial at each information sharing event. As it was discussed above, using different polynomials will result different points. However, if the externalization server performs polynomial interpolation (at $x = 0$) using its constant shares (computed based on $x_1$ and $x_2$) and the random points (computed based on the same values as well), it will result the same secret.

Submitting new points at each sharing event will preserve the privacy of the user, since the externalization server cannot trace the user in that case. However, it will allow the externalization server to verify that the user is a valid group member, if the polynomial interpolation results the group key $S$.

### 6.3.2 Implementation

In what follows, we describe the main phases of our accountable privacy-preserving scheme.

#### 6.3.2.1 Setup phase

In this phase, the authority sets up the system parameters that are going to be used in the eventual registration, authentication and tracking processes.

During the setup phase, the authority executes the following tasks:

- Define two cyclic group $G_1$ and $G_2$ of prime order $p_1$ and $p_2$.
- Define $g_1$ and $g_2$ as group generators for $G_1$ and $G_2$ respectively.
- Define a bilinear map $e' : G_1 \times G_2 \rightarrow G_T$

- Choose a random master key $S' \in Z_p^*$ and compute the group public key as: $P = e'(g_1, g_2)^{S'}$
- Choose two random values $x_1, x_2 \in Z_p^*$.
- Compute $L_1 = \frac{-x_2}{x_1 - x_2}$ and $L_2 = \frac{-x_1}{x_2 - x_1}$
- Choose a random $K \in Z_p^*$ and compute the following values:

$$T_1 = g_2^{\frac{L_1 \times L_2}{K}}, T_2 = g_2^{L_1}, T_3 = g_2^{S'} \tag{6.3}$$

$$T_4 = g_1^{\frac{K \times S'}{L_1}}, T_5 = g_2^{\frac{-1}{K \times x_1}}, T_6 = g_2^{\frac{-1}{K \times x_2}} \tag{6.4}$$

- Create the users' registry in which the authority will store the identity of any registered entity in the network. We can see the users' registry as a Hash table that maps a given key to a value.

#### 6.3.2.2 Externalization servers registration phase

In order to be able to authenticate any communicating entity in the architecture, externalization servers must request the verification parameters from the authority.

For each request coming from an externalization server, the authority executes the following tasks:

- Generate a random and unique identifier $S_{Es_i}$ for the externalization server $Es_i$.
- Send $(T_1^{S_{Es_i}}, T_2^{S_{Es_i}}, T_3^{S_{Es_i}}, T_4, T_5, T_6, P^{S_{Es_i}}, P^{S_{Es_i} \times L_1})$ to the externalization server $Es_i$.

#### 6.3.2.3 Users registration phase

Each communicating entity which wants to join the information-sharing group must perform a registration with the authority. Thus, the entity sends its digital certificate or any information that proves its identity to the authority.

Once the authority verifies the entity's identity, it performs the following operations:

- Generate a unique and random value $S_j \in Z_p^*$ specified for entity $CE_j$.
- Compute entities $CE_j$'s trace $T_j = g^{S_j}$
- Store the trace $T_j$ and entity $CE_j$'s identity in the users' registry.

- Compute the entity's share

$$CEs_i = (g_1^{\frac{S'x_1}{S_j}+\frac{S'}{L_1}}, g_1^{\frac{KS'x_2}{S_jL_1}}, A = \frac{S'}{S_jL_2}, B = \frac{S'}{S_jL_1})$$

- Send $CEs_i$ to the communicating entity $CE_j$ .

#### 6.3.2.4 Information sharing phase

In our solution, when an entity decides to share information into the fog servers, it needs to provide the digital signature and the anonymization token. This token proves that the entity is a valid group member without divulging its identity. In addition, the anonymization token links the entity to the signature provided with the shared information. In other words, it proves that the entity who signed the data is the same that provided the token. Beside the entity's anonymity and authenticity features that the token ensures at the fog servers level, it allows on the other hand the registration authority to trace communicating entities in the case of any detected misbehavior. The information sharing process in our solution works as follows:

- Choose a random value $R' \in Z_p^*$.
- Request the externalization server's public parameter $P_{Es_i} = g_2^{S_{Es_i} \times L_1}$.
- Using the shares provided by the authority and the $R'$ value, generate the anonymization token as $T = (g_1^{R'}, Y_1, Y_2)$ where

$$Y_1 = (g_1^{\frac{KS'x_2}{S_jL_1}})^{R'}$$

$$Y_2 = e'(g_1^{\frac{S'x_1}{S_j}+\frac{S'}{L_1}}, P_{Es_i})^{R'} = e'(g_1, g_2)^{\frac{R'S_{Es_i} \times S'x_1}{S_j} \times L_1 + S'R'Es_i},$$

- Generate a digital signature $Sig = (e, s)$ for data $D$ according to Schnorr scheme [Schnorr, 1989] as follows:
  1. set $r = Y_1$.
  2. Compute $e = H(r||D)$.
  3. Compute $s = R' \times (A - e \times B)$, where $A = \frac{S'}{S_jL_2}$ and $B = \frac{S'}{S_jL_1}$.
- Upload the data, its digital signature and the anonymization token into the externalization server as $(T, Sig, D)$.

We note that our solution aims to achieve a accountable privacy preserving signature scheme. Therefore, we do not consider data confidentiality service in this chapter.

#### 6.3.2.5 Authenticity and signature verification step

In order to make shared information visible for public, the externalization server starts to verify the information owner's authenticity. The authenticity verification process aims to make sure that the owner is a valid group member who could be accountable by the authority. We note that this verification process preserves the privacy of the information owner since it prevents the externalization server from discovering its identity. Besides, it does not allow to trace the owner's activity as well. Once the externalization server achieves the anonymous authenticity verification process, it also verifies that the information used to prove the authenticity of the communicating entity is related to the signature provided with the information.

The authenticity verification process runs in two steps. In the first step:

- Compute $V$ as

$$\begin{aligned} V &= e'(Y_1 \times g_1^{\frac{KS'}{L_1}}, g_2^{S_{Es_i} \times L_1 \frac{L_2}{K}}) \times Y_2 \times e'(g_1, g_2)^{S_{Es_i} S' L_1} \\ &= e'(g_1, g_2)^{\frac{S_{Es_i} S'}{S_j} L_2 (R' x_2 + S_j)} \times e'(g_1, g_2)^{\frac{S_{Es_i} S'}{S_j} L_1 (R' x_1 + S_j) + R' S' S_{Es_i}} \\ &= e'(g_1, g_2)^{\frac{S_{Es_i} S'}{S_j} \times (L_1 (R' x_1 + S_j) + L_2 (R' x_2 + S_j)) + R' S_{Es_i} S'} \end{aligned}$$

- Compute $V'$ as

$$\begin{aligned} V' &= \frac{V}{e'(g_1^{R'}, g_2^{S_{Es_i} S'})} \\ &= \frac{e'(g_1, g_2)^{\frac{S_{Es_i} S'}{S_j} \times (L_1 (R' x_1 + S_j) + L_2 (R' x_2 + S_j)) + R' S_{Es_i} S'}}{e'(g_1, g_2)^{R' S_{Es_i} S'}} \\ &= e'(g_1, g_2)^{\frac{S_{Es_i} S'}{S_j} \times (y_1 \times L_1 + y_2 \times L_2) + R' S_{Es_i} S' - R' S_{Es_i} S'} \\ &= e'(g_1, g_2)^{\frac{S_{Es_i} S'}{S_j} \times S_j} = e'(g_1, g_2)^{S_{Es_i} S'} \end{aligned}$$

  Where $y_1 = R' x_1 + S_j$ and $y_2 = R' x_2 + S_j$.

  Note that, $(y_1 \times L_1 + y_2 \times L_2 = S_j)$ represents the polynomial interpolation at $x = 0$.

  If the value $V'$ computed in this first step is equal to the externalization server

key $P = e'(g_1, g_2)^{S_{Es_i}S'}$, received from the authority, then the information owner is considered as a valid member of the information-sharing group.

Moreover, this also proves that the owner could be accountable by the authority. In fact, the externalization servers perform polynomial interpolation using values computed with $L_1$ and $L_2$. $(L_1, L_2)$ are computed using two secret values $x_1$ and $x_2$ that are known only by the authority. Therefore, in order to correctly perform the polynomial interpolation, the entity must provide shares generated using the same pieces of coordinates used to compute $(L_1, L_2)$. Since anonymization tokens, provided with the shared information, do not reveal the values $(x_1, x_2, S_i, S', K)$, the only way that allows any entity to be authenticated is to get valid credentials from the authority. As a result, a succesfull authentication means that the authority is able to trace the communicating entity.

In the second step, the externalization server proceeds to the signature verification process. This process ensures that the information owner who have provided the anonymization token is the same who signed the information.

In order to verify the signature $Sig = (e, s)$, the server executes the following steps:

1. Let $r_v = g^s$.
2. Let $e_v = H(Y_1||D)$
3. If $(e'(Y_1, g_2^{\frac{-1}{K \times x_1}}) = e'(r_v, g_2) \times e'(Y_1^{e_v}, g_2^{\frac{-1}{K \times x_2}}))$ then the signature is verified
4. Otherwise, the signature is not verified.

Note that $\frac{1}{L_2} = \frac{(x_1 - x_2)}{x_1}$ and $\frac{KS' \times x_2}{S_j \times L_1} = \frac{KS'(x_2 - x_1)}{S_j}$.

Thus, $e'(Y_1, g_2^{\frac{-1}{K \times x_2}}) = e'(g_1, g_2)^{\frac{R'S' \times (x_1 - x_2)}{S_j \times x_1}}$

$$= e'(g_1, g_2)^{\frac{R'S'}{S_j L_2}}$$

#### 6.3.2.6 Tracking step

As presented above, our solution ensures full anonymity in the externalization server. However, it also allows tracing any user, if the system detects any anomalies. With our solution, the trusted authority ensures the accountability service using the anonymization token uploaded with information. The tracking process runs as follows:

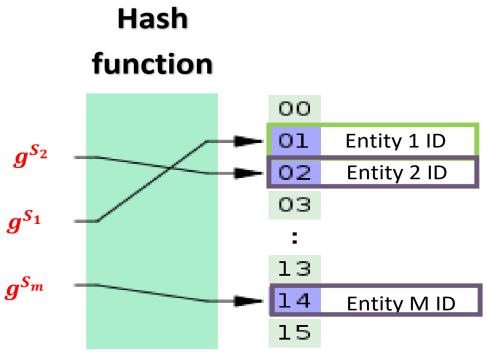



Figure 6.2 – A description of the lookup function during tracking phase

- Given the signature $Sig = (e, s)$, compute:

$$T' = \frac{s}{S' \times (\frac{1}{L_2} - \frac{1}{L_1} \times e)} = \frac{\frac{S'R'}{S_j} \times (\frac{1}{L_2} - \frac{1}{L_1} \times e)}{S' \times (\frac{1}{L_2} - \frac{1}{L_1} \times e)}$$

$$T' = \frac{R'}{S_j}$$

- Compute $T'' = (g_1^{R'})^{\frac{1}{T'}} = g^{S_j}$
- As shown in figure 6.2, the authority stores both the traces and the user's identity in a Hash Table (users' registry), it only needs to look up for $T''$ in the registry and get the corresponding user's identity.

Figure 6.3 summarizes the different steps of our solution going from the setup phase to the signature verification phase.

## 6.4 Threat model

In our protocol, we distinguish two different adversarial models where each model reflects a specific situation defined as follows:

### 6.4.1 The case where the externalization server aims to trace an entity

Let $A$ be a polynomial time adversary who interacts with a signature oracle. $A$ can submit as much arbitrary tokens as he wants to the oracle. For each received token, the oracle randomizes the token using a secure pseudo-random function (PRF) and sends it back to the adversary. Finally, the adversary outputs an arbitrary message $m$ to the oracle. The oracle chooses three random values $a, b$ and $R$ from $Z_p^*$ and

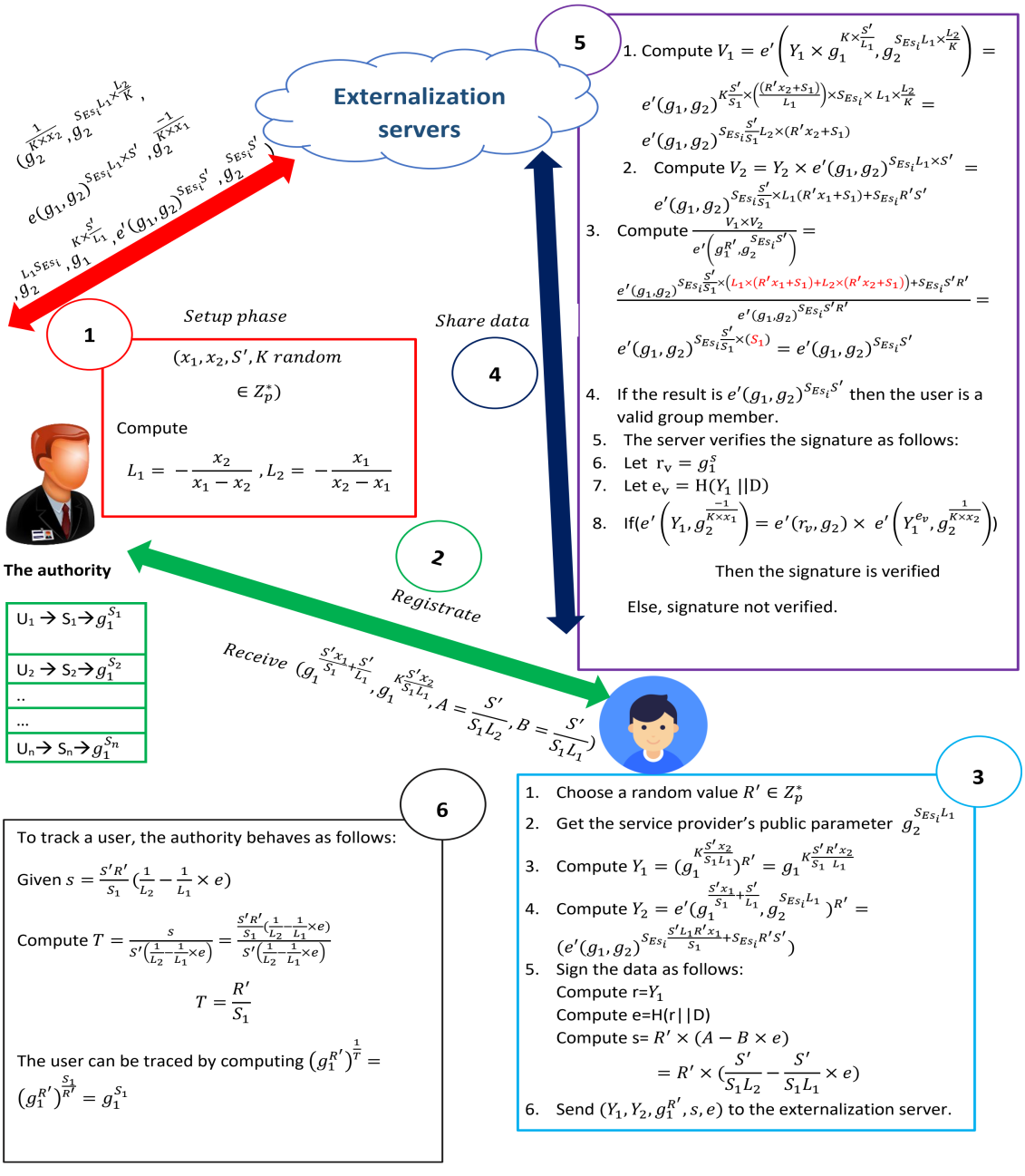



Figure 6.3 – A descriptive diagram of our protocol

sends back a randomized token $T = ((g^a)^R, (g^b)^R, g^R)$ along with the signature of the message $m$.

Adversary $A$ wins the security game if he can compute the values $g^a$ or $g^b$ given the randomized token $T$.

### 6.4.2 The case where an entity aims to forge the anonymization token

Let $A$ be a polynomial time adversary which interacts with a signature oracle. Thus, $A$ submits arbitrary messages $m_i$ to the oracle. The oracle provides the signature of these messages along with valid anonymization tokens. Finally, the adversary outputs a message $m$ that has never been submitted to the oracle along with its signature and anonymization token. Adversary $A$ wins the security game if the anonymization token along with message $m$ signature are valid.

## 6.5 Security analysis

In this section, we discuss the security of our scheme and show that it ensures the expected privacy and accountability requirements. We assume that the externalization servers are honest when it comes to authenticity verification process, so they follow the verification protocol exactly as it was described in section 6.3.

### 6.5.1 Replay/impersonation attack

An attacker may want to intercept an information signed by another entity and replays it later. To avoid that kind of situation, communicating entities should include timestamps when they share public information. In that case, it will be easy for externalization servers to detect replayed information. An attacker may also try to impersonate one of the valid communicating entities in the network. To do so, the attacker can try to generate valid credentials using brute force. Applying brute force on a cyclic group of order $p$, where $p$ is a safe prime, is a computational consuming task. Furthermore, the attacker may intercept a valid signed message and then try to extract valid credentials from token provided with the message, or to only change the message content in order to share it into the externalization server.

Extracting credentials from a signed message means that the attacker is able to guess the output of the pseudo-random function (PRF) used by the entities. Moreover, it will also require from that attacker to solve discrete logarithm problem known by its hardness in multiplicative groups. Similarly, if an attacker tries to change only

the content of the message, he will need to forge Schnorr signature. Whereas, Schnorr signature has been proved to be secure in [Pointcheval and Stern, 1996].

### 6.5.2 Privacy breach

In order to violate users' privacy in our scheme, the adversary needs to find out one of the unique values that the authority provides to each user.

Given the public information $(Y_1, Y_2, g_1^{R'}, s, e)$ made available to the adversary during each information-sharing event, we can deduce the following:

- The adversary will have no benefit from targeting the value $e$ to extract useful information since $e$ is computed based on two public values, namely $Y_2$ and the shared data $D$.
- The adversary cannot deduce any useful information from the signature $s$. In fact, $s$ is computed based on values $A$, $B$ and $e$. All these values are randomized, thus, as long as we use a secure pseudo-random function $S-PRF$, the adversary cannot distinguish between signatures randomized through $S - PRF$.

According to the deductions above, it remains in front of the adversary to use $(Y_1, Y_2, g_1^{R'})$ to breach the users' privacy. We recall that $Y_1 = (g_1^a)^{R'}$ and $Y_2 = (e'(g_1, g_2)^b)^{R'}$, where $g_1^a$ and $e'(g_1, g_2)^b$ are two values that could identify the users. Hence, the adversary will aim to trace users using either $Y_1$ or $Y_2$ along with $g_1^{R'}$.

In what follows, we will prove the security of our scheme against an adversary who tries to identify users based on $Y_1$ and $g_1^{R'}$ values. Note that the same proof can be applied on adversaries who use the $Y_2$ instead of $Y_1$ in their attack.

**Assumption 1.** *(Computational Diffie-Hellman assumption) given a multiplicative cyclic group $G$ of order $p$ with generator $g_1$, a probabilistic polynomial-time adversary has a negligible probability of computing $g_1^{ab}$ from $(g_1, g_1^a, g_1^b)$, where $a, b$ are random values in $Z_p^*$.*

**Theorem 2.** *if our scheme is broken, we can construct a polynomial time adversary who breaks assumption 1.*

*Proof.* let us call $A_1$, the adversary who breaks users' privacy in our scheme. $A_1$ plays the following security game: given $(g_1, Y_1 = (g_1^a)^{R'}, g_1^{R'})$ as input, $A_1$ tries to output $g_1^a$. If $A_1$ has a non-negligible advantage in the security game above, we can reconstruct an adversary $A_2$ which uses $A_1$ as a sub-routine, and has a non-negligible advantage in breaking assumption 1.

We recall that $A_2$ takes $(g_1, g_1^a, g_1^b)$ as inputs and tries to output $g_1^{ab}$.

The construction of $A_2$, given a polynomial adversary $A_1$ who breaks users' privacy in our scheme with a non-negligible probability, is as follows:

1. $A_2$ receives the input values $(g_1, g_1^a, g_1^b)$.
2. $A_2$ calls $A_1$ with $(g_1, g_1, g_1^b)$ as input.
3. If $A_1$ has a non-negligible advantage in breaking users' privacy in our scheme, it will output $g_1^{1/b}$.
4. $A_2$ calls $A_1$ with $(g_1, g_1^a, g_1^{1/b})$ as input.
5. If $A_1$ has a non-negligible advantage in breaking users' privacy in our scheme, it will output $g_1^{\frac{a}{1/b}} = g_1^{ab}$.
6. Finally, $A_2$ outputs $g_1^{ab}$ and breaks assumption 1.

□

**Conclusion 1.** *according to theorem 2, the existence of an adversary who breaks users' privacy in our scheme implies the existence of an adversary who breaks assumption 1. Thus, as long as assumption 1 holds our scheme is secure.*

### 6.5.3 Accountability breach

A malicious user may try to submit a token that allows him to be authenticated in the externalization servers but not to be tracked by the authority. In that kind of attacks, we can distinguish two possibilities:

In the first one, the attacker tries to generate valid credentials based on the information available in public (the anonymization tokens submitted with the shared information), without resorting to the authority. This means that the attacker needs to reveal the values $MK = (K, S', x_1, x_2)$ known only by the authority. Note that, in the values available in public, $MK$ components are protected according to the hardness of the discrete logarithm problem in multiplicative cyclic groups. Therefore, the attacker will have to solve discrete logarithm problem in order to generate valid credentials.

In the second, the attacker is a valid group member who possesses valid credentials, but he tries to modify them in a way that allows its authentication at the externalization servers but does not allow the authority to trace him. In that case, we can distinguish two possibilities: In the first possibility, the attacker combines its valid credential components in order to generate fake ones. We note

that fake credentials need to allow the user to be successfully authenticated at the externalization servers, so it needs to have the following form:

$FC = (y_1 = g_1^{\frac{S'x_1}{S_F}+\frac{S'}{L_1}}, y_2 = g_1^{\frac{KS'x_2}{S_F L_1}}, A = \frac{S'}{S_F L_2}, B = \frac{S'}{S_F L_1})$

Given the original credentials:

$OC = (y_1 = g_1^{\frac{S'x_1}{S_j}+\frac{S'}{L_1}}, y_2 = g_1^{\frac{KS'x_2}{S_j L_1}}, A = \frac{S'}{S_j L_2}, B = \frac{S'}{S_j L_1})$

We can clearly notice that the attacker has one particular challenge that consists of replacing $S_j$ by $S_F$. Faking the values $A, B$ and $y_2$ of $OC$ is an easy task. However, applying the same changes on $y_1$ requires the knowledge of $\frac{S'}{L_1}$ or $g_1^{\frac{S'}{L_1}}$. Since both values are known only by the authority, the attacker can only use brute force in order to reveal them.

In the second possibility, the attacker may collude with other users or malicious externalization servers and fake its credentials. Similarly, to the first possibility, the attacker needs to get rid of the value $\frac{S'}{L_1}$ available in $y_1$. The challenge in that case consists of finding the value $g_1^{\frac{S'}{L_1}}$ given $g^{\frac{KS'}{L_1}}$. Thus, he needs to solve discrete logarithm problem.

## 6.6 Application and performance evaluation

In this section, we apply our accountable privacy-preserving scheme on event-reporting application use case. Then, we evaluate, through simulations, its performance on the proposed use case.

Recently, several applications in relation with event prediction, recommendation systems, crowdsensing, etc. have interested researchers in both academia and industry. These applications have a major common criteria, which is events reporting. Indeed, event-reporting provides these applications with a huge amount of data, which has a direct impact on their reliability.

In event-reporting applications, we can distinguish two main data sources, which are humans and connected objects. In fact, humans could report several events such as traffic perturbations or incidents in railways, metro stations, roads, etc., through their connected objects (Smartphones, connected vehicles, etc.). Moreover, we might also have situations where connected objects report events autonomously without human interference. One can cite autonomous vehicles reporting 3D local maps; smart electricity meters reporting daily electricity consumption; parking applications where sensors report empty parking positions, etc.

It is obvious that most of the reported events will come from sources that have a direct ownership relation with humans. Therefore, it is clear that the massive

amount of events, reported a huge number of objects, is going to be very useful, but it may also leak private information about objects' owners.

Accountable and privacy-preserving are among the most important requirements to ensure while reporting events. Indeed, users cannot agree to report events by themselves or to allow their connected objects to diffuse data that may violate their privacy and expose their identities or ease tracking them. On the other hand, law authorities need to be able to ensure order and track users in case of misbehavior, which makes accountability as important as privacy-preserving.

The minimal architecture of any secure event-reporting application is composed of three main components: 1) **the users** reporting events occurring in the architecture; 2) **the authority** which manages security on the architecture and ensures accountability service if it is requested by law authorities; 3) **externalization servers** responsible for information collecting, aggregation and publishing.

Our accountable privacy-preserving solution fits perfectly with the requirements of event-reporting applications, and operates directly on its minimal architecture without requiring any additional component.

Given that most of event-reporting situations require a real time treatment, adopting fog computing paradigm becomes more suited. Thus, without loss of generality, fog nodes are going to play the role of externalization servers in our use case.

Note that the eventual event indexation and aggregation problems are not in the scope of this chapter. Moreover, we do not address in our application use case the problems related to fog computing architecture and which do not have a direct relationship with privacy-preserving and accountability.

To measure the performance of our solution, we implemented an event-reporting environment, in which:

1. we emulate the setup-launcher module (available in the authority) as a program that runs the setup phase as described in sub-section 6.3.2.1.

2. we emulate the fog-subscriber module (available in the authority) as a program that intercepts registration requests formulated by fog servers, and sends back the verification parameters as described in sub-section 6.3.2.2.

3. we emulate the subscription module (available in each fog server) as a program which requests the verification parameters from the fog-subscriber module.

4. we emulate the users-registration module (available in the authority) as a program that intercepts registration requests formulated by the users, and sends back the registration credentials as described in sub-section 6.3.2.3.

5. we emulate the registration module (available in each connected object) as a program which requests credentials from the users-registration module.

6. we emulate the event-reporting module (available in each connected object) as a program that generates and signs random events, as described in our information sharing phase (sub-section6.3.2.4), before sending it to fog servers.

7. we emulate the event-collecting module (available in each fog server) as a program that executes our signature verification algorithm (sub-section6.3.2.5) to verify the signature of the reported events, before making them available to the public.

8. we emulate the identity disclosure module as a program (available in the authority) that executes our tracking algorithm (sub-section 6.3.2.6) in order to break the anonymity of misbehaving users. This module interacts with the setup-launcher module to get some setup parameters. Moreover, it interacts with the users-registration module to get information related to users registration.

In our event-reporting environment, the authority first executes our setup-launcher module to generate the system parameters. Later on, each fog server runs its subscription module to get the verification parameters from the authority. On the other hand, each user, willing to report events in our environment, needs to call its registration module to get its registration credentials.

Once this task is successfully executed, the event-reporting module can proceed to report events. To do so, we randomly schedule a set of events to sign and report to the fog server. When public information is received, the event-collecting module uses the verification parameters, brought from the authority, to verify the signature of each reported event in order to publish it.

In the case where a misbehaving event has been pointed out to the authority, the tracking module uses the signature available in the reported event to disclose the identity of its reporter.

Figure 6.4 describes the sequence of the main actions adopted in our event-reporting environment.

To evaluate privacy-preserving and accountability features in our event-reporting environment and compares it with existing solutions, we first measure the computational time of the credential generation phase. Then, we provide comparative tables in which we compare our solution with existing accountable privacy-preserving solutions according to: 1) the number of operations performed in the credential

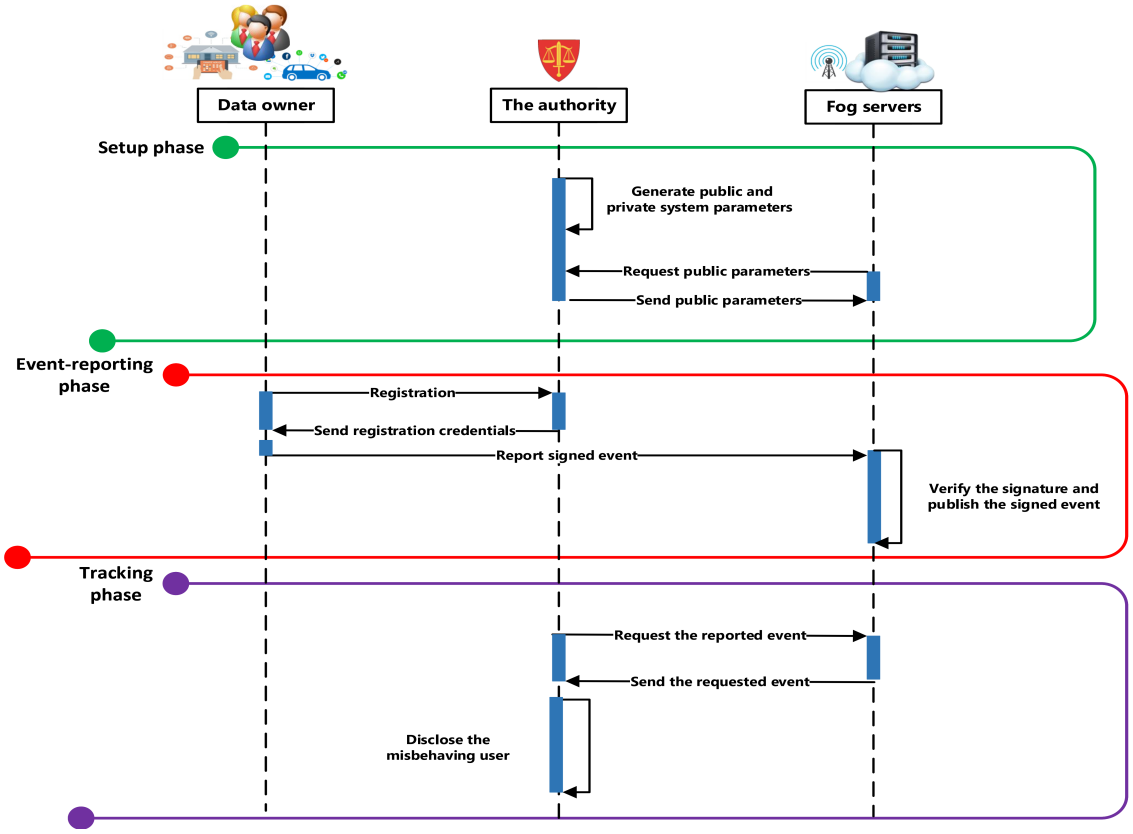



Figure 6.4 – The major sequences executed in our event-reporting environement

generation phase; 2) the number of operations performed during the signature process as well as its computational cost; 3) the signature sizes.

We also provide a comparison between our solution and existing solutions in terms of: 1) signature verification time; 2) the computational operations performed during this phase and 3) the number of computational operations performed during the tracking phase.

Finally, we simulate the arrival of reported-event requests in one fog server, and compare our solution to existing ones, according to the number of events waiting to be verified and published in that server.

We note that the experiments run on an adhoc network composed of an HP, i7 laptop with a CPU frequency of 2.7 GHZ and 16 GB of RAM, and a Toshiba i5 laptop with CPU frequency of 2.4 GHZ and 6 GB of RAM. We used pbc-0.5.14 security library in our implementation. The sizes of elements $G_1^*, G_T^*$ and $Z_p^*$ used in our implementation are $21, 61$ and 20 bytes respectively.

Moreover, we have ran 50 executions in each measurement, and the presented results represent the average of the computational time collected in these multiple executions.

### 6.6.1 Credential generation

In our scheme, all communicating entities execute the registration phase at the authority. Once the authority verifies the identity of the communicating entity, it generates a valid token that the entity will use to sign public information. Given that the authority in our scheme performs a constant number of operations in each registration, the complexity of this process is in the order of $O(n)$, where $n$ is the number of registration requests received in parallel. Table 6.1, provides a comparison between our solution and existing accountable privacy-preserving solutions, according to the number of operations performed during the credential generation phase. As shown in table 6.1, our scheme proposes a constant and less heavier credential generation process compared to existing solutions.

Tableau 6.1 – A comparative table of the computational operations performed in the credential generation phase

| Ours | Mona [Liu et al., 2013] | TPP [Wang et al., 2015] | Anonymous [Shen et al., 2018] |
|---|---|---|---|
| $2div + 2Me$ | $3P + (r + 5)Pm + 1Me$ | $2P + 4Pm + 4Me + 1Pa$ | $1P + 12kPa$ |

$(div, add)$ refer to modular division, and addition resp. $(P, Me, Pm, Pa)$ refer to paring operation, modular exponentiation, elliptic curve point multiplication and point addition resp. $(k, r)$ are two parameters defined in [Liu et al., 2013] and [Shen et al., 2018] schemes resp.

### 6.6.2 Signature process

In our event-reporting environment, all communicating entities that want to share public information through fog servers, must sign the reported events. Our signature process adopted in each event-reporting module, requires the computation of one pairing operation, three modular exponentiations, two multiplications and a Hash function. On the hand, existing solutions perform a considerable number of pairing operations, elliptic curve point multiplications (going up to eleven in the case of Mona [Liu et al., 2013]), additions and modular exponentiations in their signature process. Table 6.2, provides a comparison between our solution and the existing accountable privacy-preserving schemes in terms of the average computational cost of the signature process, signature sizes and the number of operations performed during the same process. Moreover, we show in figure 6.5, the communication overhead resulting from the transmission of signed information to the fog servers. To compute the communication overhead, we first measured the transmission time of

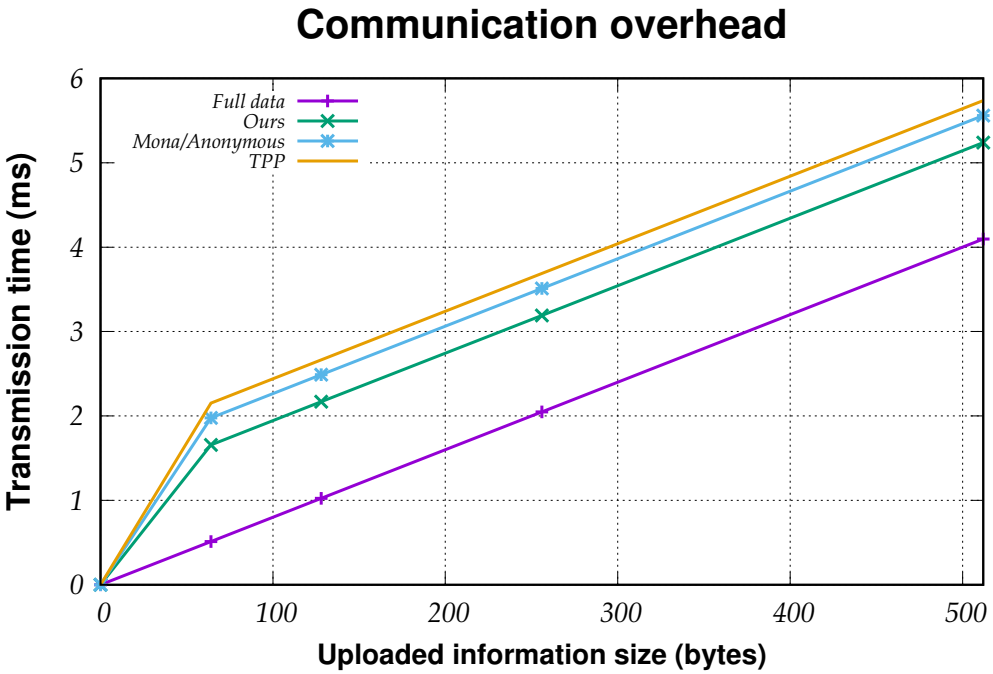



Figure 6.5 – The communication overhead resulting from the information-sharing process

Tableau 6.2 – A comparative table of the operations and computational cost of the signature and its verification phases, along with signature sizes

| | **Our** | **Mona** | **TPP** | **Anonymous** |
|---|---|---|---|---|
| Signature time (ms) | **8.55** | **33.40** | **31.13** | **33.40** |
| Signature computational operations | $1P + 3Me + 2mult + 1add$ | $11Pm + 3P + 3Me + 3Pa + 7add + 5mult$ | $4P + 3Pm + 1Pa$ | $11Pm + 3P + 3Me + 3Pa + 7add + 5mult$ |
| Signature size | $1G_T^* + 2G_1^* + 2Z_p^*$ | $3G_1^* + 6Z_p^*$ | $3G_1^* + 2G_T^* + 1Z_p^*$ | $3G_1^* + 6Z_p^*$ |
| Signature verification computational operations | $5P + 4Pm + 1Me$ | $5P + 12Pm + 6Me + 4Pa$ | $6P + 3Pm + 1Pa$ | $5P + 11Pm + 4Me + 4Pa$ |
| Verification time (ms) | **40** | **66** | **77** | **70** |

$(mult, add)$ refer to modular division, and addition resp. $(P, Me, Pm, Pa)$ refer to paring operation, modular exponentiation, elliptic curve point multiplication and point addition resp.

full data (the payload), given a network bandwidth of 10 Mbps. Then, we measured the extra transmission time induced by the signature in each scheme. As shown in figure 6.5, our solution has the lowest communication overhead since it offers the smallest signature size compared to existing solutions.

### 6.6.3 Signature verification process

In our event-reporting environment, the fog servers verify the authenticity of any reported event before making it available to the public. Figure 6.6, shows the verification time spent by the fog server to verify the authenticity of events received in parallel. As we can see, our solution outperforms existing accountable privacy-preserving solutions in terms of computational time consumed in the verification process. These results can be explained through table 6.2, where we notice that our signature verification process does not require as much point multiplications as it is required in [Shen et al., 2018] and [Liu et al., 2013]. Moreover, it does perform less pairing operations than [Wang et al., 2015].

In addition, we show in figure 6.7, a comparison between our solution and existing accountable privacy-preserving solutions according to the number of information waiting to be verified by the fog server. The results in figure 6.7 have been obtained through a simulation, in which the reporting of events follows a Poisson distribution with an arriving rate ($\lambda = \frac{1}{6}$). Thus, the fog server will receive one signed information each six milliseconds. In our simulation, each fog server defines a single Queue $Q$ that will contain signed events waiting for the signature verification process. Finally, we observe the evolution of $Q$ during the simulation time (7 seconds in our case). Figure 6.7 results show that our signature verification process achieves an average of seven reported events waiting to be verified and published along the simulation time, while Mona [Liu et al., 2013] and TPP [Wang et al., 2015] achieve an average of eleven and thirteen waiting events respectively.

### 6.6.4 Tracking process

In our event-reporting environment, the authority tracks users and reveal their identities in case of misbehavior. As shown in the benchmarks of JPBC library [JPB, ], the computational operations of the tracking process performed in our solution have more or less the same computation time as the operations performed in solutions [Liu et al., 2013, Wang et al., 2015, Shen et al., 2018], in all elliptic curve configurations. In terms of complexity, table 6.3 shows that each execution of the

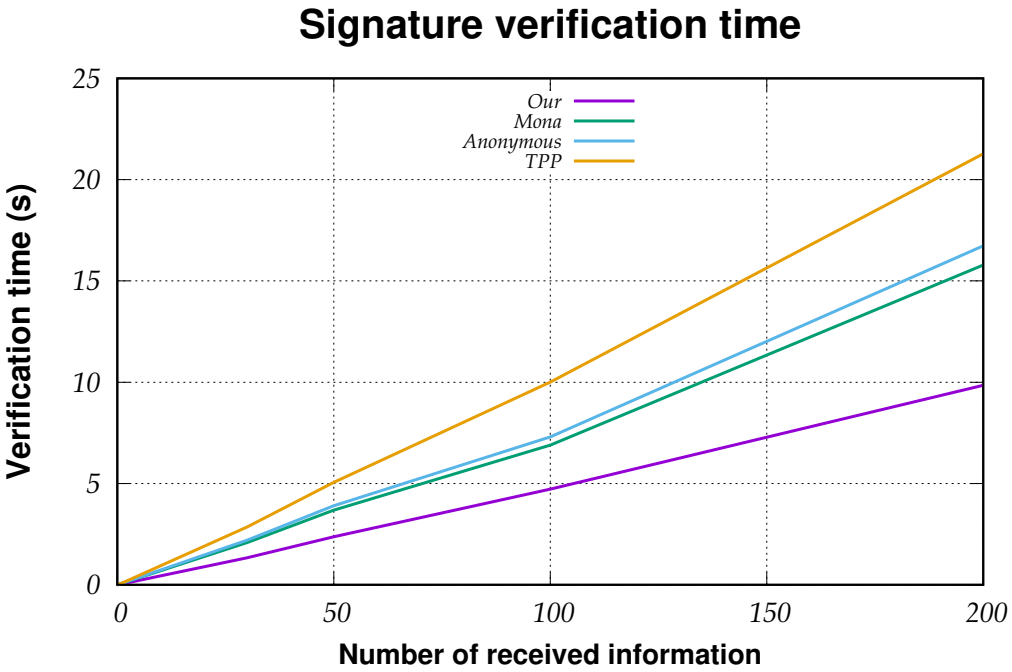



Figure 6.6 – The computational time consumed in signature verification

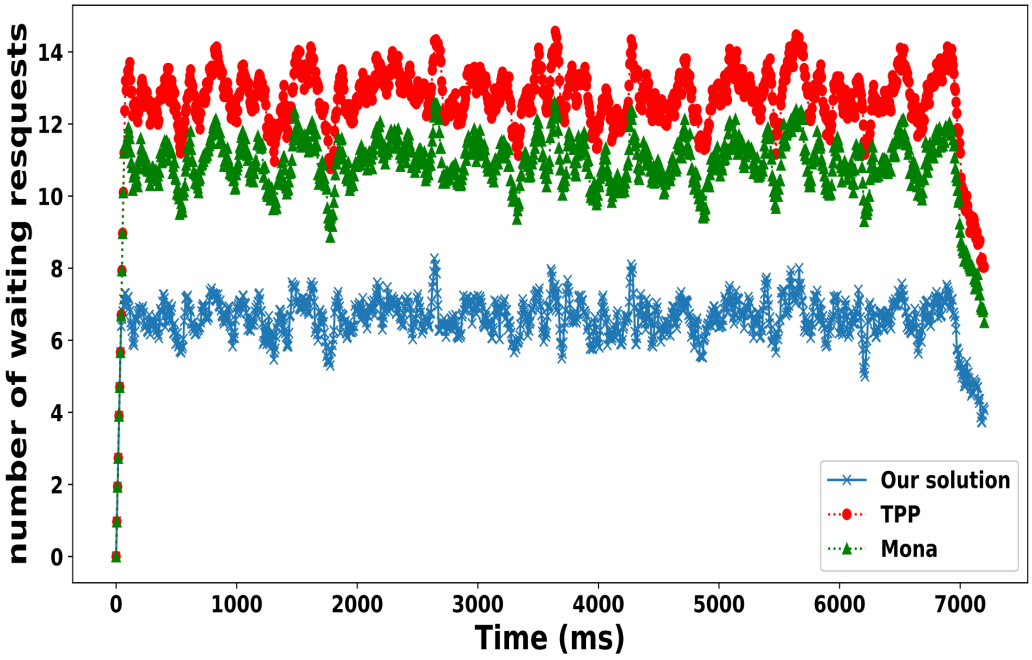



Figure 6.7 – The number of non-verified sharing requests as a function of time

tracking process in the compared solutions requires the computation of a constant number of arithmetic operations. Therefore, given $n$ tracking requests formulated in parallel to the tracking module, we can conclude that the complexity is in the order of $O(n)$.

Tableau 6.3 – A comparative table of the computational operations performed in tracking phase

| Ours | Mona [Liu et al., 2013] | TPP [Wang et al., 2015] | Anonymous [Shen et al., 2018] |
|---|---|---|---|
| $1sub + 2mult + 2div + 1Me$ | $2Pm + 2Pa$ | $2sub + 1div + 1Pm + 1Pa$ | $2Pm + 2Pa$ |

$(div, add, sub)$ refer to modular division, addition and substruction resp. $(Me, Pm, Pa)$ refer to modular exponentiation, elliptic curve point multiplication and point addition resp.

## 6.7 Conclusion

In this chapter, we have proposed a new secure, accountable privacy-preserving scheme. Based on the secret sharing method and randomization techniques, our solution allows anonymous and accountable public information sharing in information sharing architectures. In our scheme, communicating entities perform one registration with the registration authority. Then, they will be able to share information through the externalization servers without resorting to the registration authority or any third party. Each communicating entity signs the shared information with an anonymous token, which allows the externalization servers to verify the entity's authenticity without violating its privacy. In the case of anomaly detection, the authority is able to trace any communicating entity in the system, in spite of the anonymity of the provided signature. In addition to security features, our solution does not indulge a considerable overhead in terms of storage and communication. Indeed, our information-sharing process does not require several message exchanges between the servers and the communicating entities. Furthermore, externalization servers do not need to store users' pseudonyms or any temporary digital certificates. They only maintain a constant set of values that are going to allow them to verify the authenticity of any entity in the system. Besides, our scheme deals efficiently with situations where an entity tries to impersonate and share information on behalf of another one. Finally, our experimental results show that our proposal outperforms existing accountable privacy-preserving solutions.

# *Conclusion and perspectives*

## 7.1 Conclusion

Nowadays, the world is witnessing a huge expansion of Internet of Things (IoT) due to the massive growth in the number of connected devices. The huge amount of data generated by these devices require to find a proper architecture able to manage, process and store all the data.

Cloud computing is already satisfying most of the requirements needed to handle this huge technological evolution. Yet, cloud-based solutions still have some shortcomings related to real-time processing, fast data response, and latency issues. Therefore, a new architetcure, known as fog computing, which extends the cloud capabilities closer to the edge of the network has recently been introduced. However, despite the advantages offered by both architectures, many challenges still need to be resolved when adopting either one.

In this thesis, we consider data security challenges and issues in externalization technologies such as cloud or fog computing. First, we have identified in a general way, the benefits and the risks of using cloud and fog computing as a data-externalization plateform. After that, we focused our investigation on three main problems related to access control, authentication and privacy-preserving challenges.

Cryptographic access control using attribute-based encryption is one of the most efficient counter measurement that could be implemented to secure data-sharing process in cloud computing. However, this method has a serious shortcoming which is the management of revocation situations. Therefore, we first have studied the different revocation solutions proposed for attribute-based access control model in the literature. Then, we have proposed a new attribute-based access control framework with an efficient revocation method for multi-authority architectures.

Our solution ensures security requirements such as confidentiality, forward and backwardsecrecy and collusion resistance. In addition, we applied our solution on connected vehicles use case and proved its performance in terms of encryption, decryption and revocation through experimentation. Our framework provides a secure, flexible and fine-grained access control. Moreover, it deals efficiently with the

revocation problem without launching any key regeneration process and performing any changes on the users' side. Furthermore, unlike most of the existing solutions, the authorities are not the only entities responsible for the revocation in our scheme. Henceforth, data owner are able to set up a personal data sharing domain and manage the revocation in their own domain without relying on any third party. We go farther in our solution and provide a new efficient revocation mode for the fully distributed data-sharing model.

Later on, we have investigated authentication problems in fog computing environement. Our study concluded that adopting fog architecture requires some specific features in the authentication process. One can cite the mutual authentication feature, the interoperability and component heterogenity support, low latency, dynamicity and scallability support. It was obvious that it is not sufficient to directly adopt traditional authentication mechanisms based on certificates, passwords and biometric definition on fog computing architecture. Therefore, we have proposed a new secure authentication scheme based on secret sharing and blockchain technology to manage authentication in fog computing architecture.

In our scheme, both the users and the fog nodes perform one registration in the cloud level. Then, they will be able to mutually authenticate each other at the edge of the network without resorting to the cloud. The users are able to verify the authenticity of any legitimate fog. Moreover, fog nodes in our solution do not need to store any users' identifiers and any digital certificates; they only hold a couple of values that are going to allow them to verify the authenticity of any user in the system. In addition, fog nodes can also authenticate each other at the edge of the network using the blockchain. Furthermore, our scheme deals efficiently with situations where an entity from the system tries to impersonate another one in order to get services from fog nodes.

Finally, we tackeled accountablility and privacy-preserving challenge in information-sharing plateforms such as cloud and fog computing. In fact, sharing information into cloud or fog servers may drive service providers to use users' personal information for business purposes in most of the cases. It is true that this information allow the providers to enhance their services, but it also represents a valuable source of benefits if it is selled to advertising companies. Selling personal information clearly violates users' privacy since it is usually done without users complete approval. Therefore, we first conducted a deep review on research papers that addressed privacy issues in information-sharing applications. From our investigation, we noticed that many contributions tend to anonymazation techniques to preserve users' privacy, but few of them considered accountability service. However, full anonymity may encourage users to act maliciously and thus,

it is necessary to provide accountability mechanisms.

Consequently, we have proposed a new secure, accountable and privacy-preserving scheme for information-sharing applications. Our solution allows the externalization servers to verify users' authenticity without violating their privacy. However, in the case of anomaly detection, we consider an authority that is able to trace any communicating entity in the system, in spite of its anonymity. Besides, our scheme deals is robust against authetication credentials forgery and impersonation attacks. In addition to security features, our solution does not indulge a considerable overhead in terms of storage and communication. Indeed, our information-sharing process does not require several exchanges between the servers and the communicating entities. Furthermore, by only holding a constant set of values, any externalization server in the architecture is able to verify the authenticity of any entity in the system.

## 7.2 Perspectives

We indetified two main directions for our future works. First, we intend to address the revocation challenge in our mutual authentication scheme that we proposed for fog computing architecture. A simple approach to manage revocation in our proposed solution would rely on revocation list to store users credentials. However, this approach has several limitations. One can cite, the management of list update process which is not a simple task due to the huge number of active entities in fog computing architecture. In addition, in case of high rate of revocations the list size will considerably increases. Moreover, the revocation in our proposed scheme is not limited to users, but it also includes the revocation of fog nodes. Therefore, if a fog node has been revoked, we need to ensure that the users get this information by being able to have access to the latest transactions inserted in the blockchain.

Besides, we also intend to address the problem of conditional revocation in our accountable privacy-preserving scheme. The conditionnal revocation concept means that the authority needs to provide mechanisms for temporary or permanently prevent malicious users from sharing public information. To illustrate the need of a such mechanism let us give two example:

Suppose that a user shares a false information in the network through its mobile phone. In that case, the authority will detect this misbehavior and tack the user's identity in order to proceed to judicial follow-up. Till this end, the user need to be prevented from sharing public information. However, the temporary revocation in this situation is most logical and reliable choice since it's neither fair nor efficient for our information sharing system to revock users pemanently as soon as they

misbehave.

Now suppose that a user reports to the authority the theft of its mobile phone that he usually use to share public information. Consequently, the authority is supposed to permanently revock the user's credential in that case. Otherwise, he might take responsability of any misbehaving action that the thief may perform using the stolen victim's mobile.

In both cases, the revocation remains a challenge in our solution because of privacy-preserving feature that any proposed revocation mechanism has to maintain. Moreover, the proposed mechanisms should support scalability and preferably avoid highly frequent credential updates that the legitimate users may endure.

# *Bibliography*


[Ama, ] Aws cloudsplains what happened during s3 storage outage | techcrunch. [Online], 28 décembre 2018 [consulted on 28 décembre 2018].

[con, ] Connected car report 2016: Opportunities, risk, and turmoil on the road to autonomous, https://www.strategyand.pwc.com/reports/connected-car-2016-study. 29 january 2018 [consulted on 29 january 2018].

[Cyb, ] Cyber attacks briefly knock out top sites. [Online], 28 décembre 2018 [consulted on 28 décembre 2018].

[dro, ] Dropbox, https://www.dropbox.com/. [online].Dropbox, 29 january 2018 [consulted on 29 january 2018].

[JPB, ] Jpbc-java pairing-based cryptography library: Benchmark, http://gas.dia.unisa.it/projects/jpbc/benchmark.html. 4 december 2013 [consulted on 20 may 2019].

[Pol, ] La police de dubaï passe à la voiture autonome, https://humanoides.fr/dubai-voiture-autonome/. 29 january 2018 [consulted on 29 january 2018].

[dub, ] La première voiture de police autonome patrouillera bientôt à dubaï, https://sciencepost.fr/2017/07/premiere-voiture-de-police-autonome-patrouillera-bientot-a-dubai/. 29 january 2018 [consulted on 29 january 2018].

[rob, ] Rise of the robocar: are connected cars safer, or a target for hackers? | technology | the guardian, https://www.theguardian.com/technology/2017/aug/13/robot-connected-cars-hacking-risks-driverless-vehicles-ross-now. 29 january 2018 [consulted on 29 january 2018].

[azu, ] Services et plateforme de cloud computing microsoft azure, https://azure.microsoft.com/fr-fr/. [Online], 29 january 2018 [consulted on 29 january 2018].

[rig, ] state of the cloud report. [Online], 1 mars 2018 [consulted on 28 decembre 2018].

[ama, 2018] ("2018"). Amazon web services (aws) - cloud computing services, https://aws.amazon.com/fr. [Online].aws, 18 january 2018 [consulted on 29 january 2018].

[Cis, 2018] ("2018"). Cisco visual networking index: Forecast and methodology, 2016–2021 - complete-white-paper-c11-481360.pdf. 6 november 2018 [consulted on 6 november 2018].

[Git, 2018] ("2018"). Github, [online], https://github.com/ethereum/go-ethereum. [Online], 18 january 2018 [consulted on 29 january 2018].

[Alaba et al., 2017] Alaba, F. A., Othman, M., Hashem, I. A. T., and Alotaibi, F. (2017). Internet of things security: A survey. *Journal of Network and Computer Applications*, 88:10–28.

[Alrawais et al., 2017] Alrawais, A., Alhothaily, A., Hu, C., and Cheng, X. (2017). Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Computing*, 21(2):34–42.

[Aslam and Zou, 2009] Aslam, B. and Zou, C. (2009). Distributed certificate and application architecture for vanets. In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1–7. IEEE.

[Attrapadung and Imai, 2009] Attrapadung, N. and Imai, H. (2009). Conjunctive broadcast and attribute-based encryption. In *International Conference on Pairing-Based Cryptography*, pages 248–265. Springer.

[Balfanz et al., 2002] Balfanz, D., Smetters, D. K., Stewart, P., and Wong, H. C. (2002). Talking to strangers: Authentication in ad-hoc wireless networks. In *NDSS*. Citeseer.

[Barona and Anita, 2017] Barona, R. and Anita, E. M. (2017). A survey on data breach challenges in cloud computing security: Issues and threats. In *Circuit,*

*Power and Computing Technologies (ICCPCT), 2017 International Conference on*, pages 1–8. IEEE.

[Bethencourt et al., 2007] Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 321–334. IEEE.

[Bishop and Gates, 2008] Bishop, M. and Gates, C. (2008). Defining the insider threat. In *Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead*, page 15. ACM.

[Blakley, 1979] Blakley, G. (1979). Safeguarding cryptographic keys 48 proceedings of the national computer conference. In *AFIPS Conference Proceedings June*.

[Boneh and Franklin, 2001] Boneh, D. and Franklin, M. (2001). Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer.

[Booth et al., 2013] Booth, G., Soknacki, A., and Somayaji, A. (2013). Cloud security: Attacks and current defenses. In *8th Annual symposium on information Assurance (ASIA'13)*, pages 4–5.

[Borgh et al., 2017] Borgh, J., Ngai, E., Ohlman, B., and Malik, A. M. (2017). Employing attribute-based encryption in systems with resource constrained devices in an information-centric networking context. In *Global Internet of Things Summit (GIoTS), 2017*, pages 1–6. IEEE.

[Bouzefrane et al., 2014] Bouzefrane, S., Mostefa, A. F. B., Houacine, F., and Cagnon, H. (2014). Cloudlets authentication in nfc-based mobile computing. In *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on*, pages 267–272. IEEE.

[Brainard et al., 2006] Brainard, J., Juels, A., Rivest, R. L., Szydlo, M., and Yung, M. (2006). Fourth-factor authentication: somebody you know. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 168–178. ACM.

[Brunette et al., 2009] Brunette, G., Mogull, R., et al. (2009). Security guidance for critical areas of focus in cloud computing v2. 1. *Cloud Security Alliance*, pages 1–76.

[Chase, 2007] Chase, M. (2007). Multi-authority attribute based encryption. In *Theory of Cryptography Conference*, pages 515–534. Springer.

[Christidis and Devetsikiotis, 2016] Christidis, K. and Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *Ieee Access*, 4:2292–2303.

[Cramer and Shoup, 2000] Cramer, R. and Shoup, V. (2000). Signature schemes based on the strong rsa assumption. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):161–185.

[Cui et al., 2018] Cui, J., Zhou, H., Zhong, H., and Xu, Y. (2018). Akser: Attribute-based keyword search with efficient revocation in cloud computing. *Information Sciences*, 423:343–352.

[De and Ruj, 2017] De, S. J. and Ruj, S. (2017). Efficient decentralized attribute based access control for mobile clouds. *IEEE Transactions on Cloud Computing*.

[Dukes, 2015] Dukes, C. (2015). Committee on national security systems (cnss) glossary. [Online], 26 avril 2018[consulted on 29 january 2018].

[Erika McCallister, 2010] Erika McCallister, Timothy Grance, K. A. S. ("2010"). Guide to protecting the confidentiality of personally identifiable information (pii). [Online], 7 janvier 2019 [consulted on 29 january 2018].

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

[Hogan and Sokol, ] Hogan, M. and Sokol, A. Nist cloud computing standards roadmap. version 2. nist cloud computing standards roadmap working group.

[Hu et al., 2017] Hu, P., Dhelim, S., Ning, H., and Qiu, T. (2017). Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications*, 98:27–42.

[Huang et al., 2015] Huang, X., Tao, Q., Qin, B., and Liu, Z. (2015). Multi-authority attribute based encryption scheme with revocation. In *Computer Communication and Networks (ICCCN), 2015 24th International Conference on*, pages 1–5. IEEE.

[Hur and Noh, 2011] Hur, J. and Noh, D. K. (2011). Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1214–1221.

[Ibrahim, 2016] Ibrahim, M. H. (2016). Octopus: An edge-fog mutual authentication scheme. *IJ Network Security*, 18(6):1089–1101.

[Ibraimi et al., 2009] Ibraimi, L., Petkovic, M., Nikova, S., Hartel, P., and Jonker, W. (2009). Mediated ciphertext-policy attribute-based encryption and its application. In *International Workshop on Information Security Applications*, pages 309–323. Springer.

[Jaeger and Schiffman, 2010] Jaeger, T. and Schiffman, J. (2010). Outlook: Cloudy with a chance of security challenges and improvements. *IEEE Security & Privacy*, 8(1).

[Jansen and Grance, 2011] Jansen, W. and Grance, T. (2011). Sp 800-144. guidelines on security and privacy in public cloud computing.

[Ji et al., 2016] Ji, S., Mittal, P., and Beyah, R. (2016). Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey. *IEEE Communications Surveys & Tutorials*, 19(2):1305–1326.

[Junod and Karlov, 2010] Junod, P. and Karlov, A. (2010). An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. In *Proceedings of the tenth annual ACM workshop on Digital rights management*, pages 13–24. ACM.

[Kandias et al., ] Kandias, M., Virvilis, N., and Gritzalis, D. The insider threat in cloud computing.

[Keshavarz and Anwar, 2018] Keshavarz, M. and Anwar, M. (2018). Towards improving privacy control for smart homes: A privacy decision framework. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–3. IEEE.

[Kumar, 2010a] Kumar, M. (2010a). An enhanced remote user authentication scheme with smart card. *IJ Network Security*, 10(3):175–184.

[Kumar, 2010b] Kumar, M. (2010b). A new secure remote user authentication scheme with smart cards. *IJ Network Security*, 11(2):88–93.

[Leavitt, 2011] Leavitt, N. (2011). Internet security under attack: The undermining of digital certificates. *Computer*, 44(12):17–20.

[Lewko et al., 2010] Lewko, A., Sahai, A., and Waters, B. (2010). Revocation systems with very small private keys. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 273–285. IEEE.

[Lewko and Waters, 2011] Lewko, A. and Waters, B. (2011). Decentralizing attribute-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 568–588. Springer.

[Liu et al., 2017] Liu, H., Zhu, P., Chen, Z., Zhang, P., and Jiang, Z. L. (2017). Attribute-based encryption scheme supporting decryption outsourcing and attribute revocation in cloud storage. In *Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), 2017 IEEE International Conference on*, volume 1, pages 556–561. IEEE.

[Liu et al., 2012] Liu, J. K., Au, M. H., Susilo, W., and Zhou, J. (2012). Enhancing location privacy for electric vehicles (at the right time). In *European Symposium on Research in Computer Security*, pages 397–414. Springer.

[Liu et al., 2013] Liu, X., Zhang, Y., Wang, B., and Yan, J. (2013). Mona: Secure multi-owner data sharing for dynamic groups in the cloud. *ieee transactions on parallel and distributed systems*, 24(6):1182–1191.

[Los et al., 2013] Los, R., Shackleford, D., and Sullivan, B. (2013). The notorious nine cloud computing top threats in 2013. *Cloud Security Alliance*.

[Lu et al., 2014] Lu, N., Cheng, N., Zhang, N., Shen, X., and Mark, J. W. (2014). Connected vehicles: Solutions and challenges. *IEEE internet of things journal*, 1(4):289–299.

[Lu et al., 2008] Lu, R., Cao, Z., Chai, Z., and Liang, X. (2008). A simple user authentication scheme for grid computing. *IJ Network Security*, 7(2):202–206.

[Luis T. A. N. Brandao, 2018] Luis T. A. N. Brandao, Nicky Mouha, A. V. ("2018"). Draft nistir 8214, threshold schemes for cryptographic primitives: Challenges and opportunities in standardization and validation of threshold cryptography - nistir-8214-draft.pdf. [Online], 26 avril 2018[consulted on 29 january 2018].

[McIntosh and Austel, 2005] McIntosh, M. and Austel, P. (2005). Xml signature element wrapping attacks and countermeasures. In *Proceedings of the 2005 workshop on Secure web services*, pages 20–27. ACM.

[Mell and Grance, 2011] Mell, P. and Grance, T. ("2011"). Sp 800-145, the nist definition of cloud computing | csrc. [Online].nist, 21 décembre 2018[consulted on 21 december 2018].

[Michaela Iorga, 2018] Michaela Iorga, Nedim Goren, L. F. R. B. M. J. M. C. M. ("2018"). Fog computing conceptual model: Recommendations of the national institute of standards and technology - nist.sp.500-325.pdf. [Online], 26 avril 2018[consulted on 29 january 2018].

[Modi et al., 2013] Modi, C., Patel, D., Borisaniya, B., Patel, A., and Rajarajan, M. (2013). A survey on security issues and solutions at different layers of cloud computing. *The journal of supercomputing*, 63(2):561–592.

[Muller et al., 2009] Muller, S., Katzenbeisser, S., and Eckert, C. (2009). On multi-authority ciphertext-policy attribute-based encryption. *Bulletin of the Korean Mathematical Society*, 46(4):803–819.

[Nakamoto, 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[Neuman and Ts'o, 1994] Neuman, B. C. and Ts'o, T. (1994). Kerberos: An authentication service for computer networks. *IEEE Communications magazine*, 32(9):33–38.

[Nicanfar et al., 2013] Nicanfar, H., Hosseininezhad, S., TalebiFard, P., and Leung, V. C. (2013). Robust privacy-preserving authentication scheme for communication between electric vehicle as power energy storage and power stations. In *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, pages 55–60. IEEE.

[Ostrovsky et al., 2007] Ostrovsky, R., Sahai, A., and Waters, B. (2007). Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 195–203. ACM.

[Panayappan et al., 2007] Panayappan, R., Trivedi, J. M., Studer, A., and Perrig, A. (2007). Vanet-based approach for parking space availability. In *Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*, pages 75–76. ACM.

[Pirretti et al., 2010] Pirretti, M., Traynor, P., McDaniel, P., and Waters, B. (2010). Secure attribute-based systems. *Journal of Computer Security*, 18(5):799–837.

[Pointcheval and Stern, 1996] Pointcheval, D. and Stern, J. (1996). Security proofs for signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 387–398. Springer.

[Rigney et al., 2000] Rigney, C., Willens, S., Rubens, A., and Simpson, W. (2000). Remote authentication dial in user service (radius). Technical report.

[Rivas et al., 2011] Rivas, D. A., Barceló-Ordinas, J. M., Zapata, M. G., and Morillo-Pozo, J. D. (2011). Security on vanets: Privacy, misbehaving nodes, false information and secure data aggregation. *Journal of Network and Computer Applications*, 34(6):1942–1955.

[Rivest et al., 1978] Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

[Roman et al., 2018] Roman, R., Lopez, J., and Mambo, M. (2018). Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 78:680–698.

[Rottondi et al., 2014a] Rottondi, C., Fontana, S., and Verticale, G. (2014a). Enabling privacy in vehicle-to-grid interactions for battery recharging. *Energies*, 7(5):2780–2798.

[Rottondi et al., 2014b] Rottondi, C., Fontana, S., and Verticale, G. (2014b). A privacy-friendly framework for vehicle-to-grid interactions. In *International Workshop on Smart Grid Security*, pages 125–138. Springer.

[Ruj et al., 2011] Ruj, S., Nayak, A., and Stojmenovic, I. (2011). Dacc: Distributed access control in clouds. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 91–98. IEEE.

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

[Salem et al., 2010] Salem, F. M., Ibrahim, M. H., and Ibrahim, I. (2010). Non-interactive authentication scheme providing privacy among drivers in vehicle-to-vehicle networks. In *Networking and Services (ICNS), 2010 Sixth International Conference on*, pages 156–161. IEEE.

[Schnorr, 1989] Schnorr, C.-P. (1989). Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*, pages 239–252. Springer.

[Shamir, 1979] Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11):612–613.

[Shen et al., 2018] Shen, J., Zhou, T., Chen, X., Li, J., and Susilo, W. (2018). Anonymous and traceable group data sharing in cloud computing. *IEEE Transactions on Information Forensics and Security*, 13(4):912–925.

[Shirey, 2007] Shirey, R. (2007). Internet security glossary, version 2. Technical report.

[Singh and Chatterjee, 2017] Singh, A. and Chatterjee, K. (2017). Cloud security issues and challenges: A survey. *Journal of Network and Computer Applications*, 79:88–115.

[Singh et al., 2016] Singh, S., Jeong, Y.-S., and Park, J. H. (2016). A survey on cloud computing security: Issues, threats, and solutions. *Journal of Network and Computer Applications*, 75:200–222.

[Staddon et al., 2008] Staddon, J., Golle, P., Gagné, M., and Rasmussen, P. (2008). A content-driven access control system. In *Proceedings of the 7th symposium on Identity and trust on the Internet*, pages 26–35. ACM.

[Sweeney, 2002] Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570.

[Talbot, 2009] Talbot, D. (2009). Vulnerability seen in amazon's cloud-computing. *MIT Tech Review*.

[Vincent C. Hu, 2014] Vincent C. Hu, David Ferraiolo, R. K. A. S. K. S. R. M. K. S. ("2014"). Guide to attribute based access control (abac) definition and considerations - nist.sp.800-162.pdf. [Online], 26 avril 2018[consulted on 29 january 2018].

[Wang et al., 2015] Wang, H., Qin, B., Wu, Q., Xu, L., and Domingo-Ferrer, J. (2015). Tpp: Traceable privacy-preserving communication and precise reward for vehicle-to-grid networks in smart grids. *IEEE Transactions on Information Forensics and Security*, 10(11):2340–2351.

[Whaiduzzaman et al., 2014] Whaiduzzaman, M., Sookhak, M., Gani, A., and Buyya, R. (2014). A survey on vehicular cloud computing. *Journal of Network and Computer Applications*, 40:325–344.

[Wichers, ] Wichers, D. The top 10 most critical web application security risks. [Online], 11 décembre 2011 [consulted on 28 décembre 2018].

[Xu and Martin, 2012] Xu, Z. and Martin, K. M. (2012). 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. IEEE.

[Yan and Shi, 2017] Yan, Z. and Shi, W. (2017). Cloudfile: A cloud data access control system based on mobile social trust. *Journal of Network and Computer Applications*, 86:46–58.

[Yang and Jia, 2014] Yang, K. and Jia, X. (2014). Expressive, efficient, and revocable data access control for multi-authority cloud storage. *IEEE transactions on parallel and distributed systems*, 25(7):1735–1744.

[Yang et al., 2012] Yang, K., Liu, Z., Cao, Z., Jia, X., Wong, D. S., and Ren, K. (2012). Taac: Temporal attribute-based access control for multi-authority cloud storage systems. *IACR Cryptology EPrint Archive*, 2012:651.

[Yu et al., 2010] Yu, S., Wang, C., Ren, K., and Lou, W. (2010). Attribute based data sharing with attribute revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 261–270. ACM.

[Zhong et al., 2018] Zhong, H., Zhu, W., Xu, Y., and Cui, J. (2018). Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage. *Soft Computing*, 22(1):243–251.