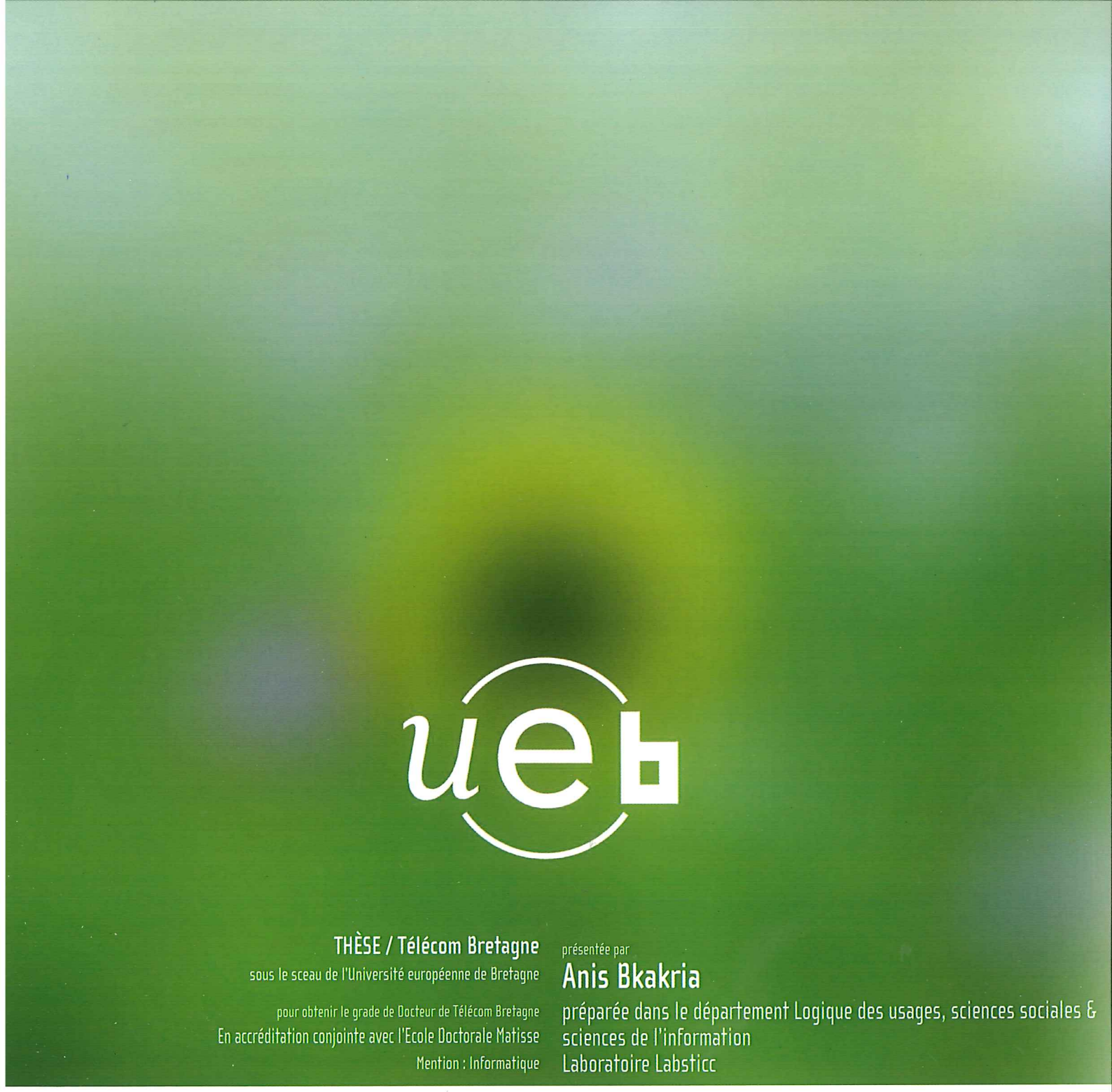
Specification and deployment of integrated Security Policy for Outsourced Data

Anis Bkakria



THÈSE / Télécom Bretagne
sous le sceau de l'Université européenne de Bretagne

pour obtenir le grade de Docteur de Télécom Bretagne
En accréditation conjointe avec l'Ecole Doctorale Matisse
Mention : Informatique

présentée par
Anis Bkakria
préparée dans le département Logique des usages, sciences sociales & sciences de l'information
Laboratoire Labsticc

Specification and Deployment of Integrated Security Policy for Outsourced Data

Thèse soutenue le 17 décembre 2015
Devant le jury composé de :

Refik Molva
Professeur, Eurecom - Sophia Antipolis / président

Régine Laleau
Professeur, Université Paris-Est Créteil / rapporteur

Michaël Rusinowitch
Directeur de Recherche, Loria-Inria Lorraine – Villiers les Nancy / rapporteur

Nora Cuppens-Boulahia
Directrice de recherche, Télécom Bretagne / examinateur

David Gross-Amblard
Professeur, Université de Rennes 1 / examinateur

Louis Rilling
Ingénieur , DGA Maîtrise de l'information - Rennes / examinateur

Pascal Lafourcade
Maître de conférences, IUT Clermont-Ferrand - Université d'Auvergne / examinateur

Frédéric Cuppens
Professeur, Télécom Bretagne / directeur de thèse

N° d'ordre : 2015telb0376

Thèse

Présentée à

Télécom Bretagne

en habilitation conjointe avec l'Université de Rennes I

pour obtenir le grade de
Docteur de Telecom Bretagne

Mention : *Informatique*

par

Anis Bkakria

Specification and Deployment of Integrated Security Policy for Outsourced Data

---

soutenue le 17 décembre 2015

Composition du Jury :

Président :	- Refik Molva, Professeur, Eurecom
Rapporteurs :	- Régine Laleau, Professeur, Université Paris-Est Créteil - Michaël Rusinowitch, Directeur de Recherche, LORIA-INRIA
Examinateurs :	- Frédéric Cuppens, Professeur, Télécom Bretagne Rennes - Nora Cuppens , Directeur de recherche, Télécom Bretagne Rennes - David Gross-Amblard, Professeur, Université de Rennes 1 & IRISA - Pascal Lafourcade, Maître de conférences, LIMOS, Université Clermon Auvergne - Louis Rilling, Ingénieur Chercheur, DGA Maîtrise de l'information

Dedicated to my parents, my wife, my two sisters, and my brother for ***1001*** *reasons.*

Abstract

Recent advance in cloud computing has transformed the way information is managed and consumed, since this new paradigm provides cost efficient solutions that allow the transmission, storage, and intensive computing of information. Therefore, Cloud service providers are increasingly required to take responsibility for the storage as well as the efficient and reliable sharing of information, thus carrying out a "data outsourcing" architecture. Despite that outsourcing information on Cloud service providers may cut down data owners' responsibility of managing data while increasing data availability, data owners hesitate to fully trust Cloud service providers to protect their outsourced data. Recent data breaches on Cloud storage providers have exacerbated these security concerns. In response, security designers defined approaches that provide high level security assurance, such as encrypting data before outsourcing them to Cloud servers. Such traditional approaches bring however the disadvantage of prohibiting useful information release (e.g., efficient querying of outsourced data). This raises then the need to come up with new models and approaches for defining and enforcing security and utility policies on outsourced data.

This thesis aims to address this trade-off, while considering two kind of security policies.

In the first hand, we focus on confidentiality policies specification and enforcement, which requires enforcing the secrecy of outsourced data stored by an untrusted Cloud service provider, while providing an efficient use (e.g., searching and computing) of the outsourced data by different authorized users. To this end, first, we proposed an approach ensuring the confidentiality of sensitive information in outsourced multi-relational databases by combining data fragmentation and encryption techniques. We then defined a secure and effective querying method for data hosted on several service providers. Afterwards, we improved the security of the querying technique we defined in order to protect data confidentiality under a collaborative Cloud storage service providers model. Second, We defined a policy-based configuration framework for sensitive outsourced data allowing the data owner to specify the confidentiality re-

quirements (e.g., the set of sensitive information) and utility requirements (e.g., SQL queries that should be executed over the encrypted data) over the data to be outsourced. We then proposed an efficient technique to find a combination of encryption schemes that satisfies a near optimal trade-off between confidentiality requirements and utility requirements.

On the other hand, we address the problem of heterogeneous security policies (e.g., confidentiality requirements, privacy requirements, ownership requirements, etc) specification and deployment. First, we defined an expressive formal language allowing to specify as finely as possible heterogeneous security and utility requirements over the data structure to be outsourced, as well as existing security mechanisms that can be used to enforce them. Second, we defined a reasoning method for our formal model allowing outsourced data owners to automatically get the combination of security mechanisms providing a near optimal trade-off between the security and the utility of the data to be outsourced and the complexity of its application over the used system. Finally, we used a data tainting method to extend our previously defined reasoning method. That is, in cases in which no combination of security mechanisms could fully satisfy the chosen heterogeneous policy, our improved reasoning method figure out a combination of security mechanisms ensuring the best compromise between security requirements to be enforced and the set of utility requirements to satisfy over the outsourced data.

Résumé

Les avancées récentes dans les domaines de l'informatique en nuage sont en train de transformer la manière dont les informations sont gérées et consommées. Ceci est dû au fait que ce nouveau paradigme offre des solutions rentables permettant, la transmission, le stockage, et le calcul intensif des informations. Par conséquent, les fournisseurs de services de l'informatique en nuage sont de plus en plus tenus d'assumer leur responsabilité par rapport au stockage, ainsi que du partage efficace et fiable des données externalisées. En effet, bien qu'ils soient convaincus que l'externalisation des données puisse contribuer à réduire leurs responsabilités relativement à la gestion des données et à augmenter la disponibilité de leurs données, les propriétaires des données hésitent à accorder une confiance aveugle aux fournisseurs de services Cloud quant à la protection de leurs données externalisées. Ce manque de confiance est dû au fait que, pour les récentes fuites de données externalisées, des vulnérabilités au sein des services Cloud ont été exploitées. En réponse à ces préoccupations, plusieurs approches ont été définies afin d'assurer un niveau élevé de sécurité, telles que le chiffrement des données avant leur externalisation. De telles approches traditionnelles apportent toutefois l'inconvénient d'empêcher une utilisation efficace des données externalisées (par exemple, interroger une base de données externalisées). Cela soulève alors la nécessité d'inventer de nouveaux modèles et de nouvelles approches permettant la définition et la mise en application des politiques de sécurité et de fonctionnalité sur les données externalisées.

Cette dissertation vise à surmonter ce dilemme, tout en tenant compte de deux types de politiques de sécurité.

En premier lieu, nous nous concentrons sur la spécification et le déploiement des politiques de confidentialité, qui nécessite : (1) la protection des données sensibles externalisées stockées dans des serveurs Cloud considérés comme étant non fiables, et (2) l'utilisation efficace des données externalisées par les utilisateurs autorisés. Pour répondre à cette préoccupation, nous avons, d'une part, proposé une approche combinant la fragmentation et le chiffrement des données afin d'assurer la confidentialité des

informations sensibles stockées dans des bases de données multi-relationnelles. Nous avons ensuite défini une méthode sûre et efficace permettant l'interrogation des données hébergées sur des fragments distribués dans différents fournisseurs de services Cloud. Finalement, nous avons amélioré notre approche avec l'utilisation d'une technique d'interrogation de données basée sur le retrait d'informations privé (PIR) afin d'assurer la confidentialité des données dans le cas de collaboration entre les fournisseurs de services. D'autre part, nous avons défini une solution permettant aux propriétaires des données de spécifier leurs besoins de confidentialité (par exemple, les données sensibles à protéger) ainsi que leurs besoins de fonctionnalité (par exemple, les requêtes SQL qui doivent être exécutées) sur les données externalisées. Notre solution implémente une technique efficace pour trouver la combinaison de mécanismes de chiffrement permettant de satisfaire le meilleur compromis entre la sécurité et l'utilité des données externalisées.

En deuxième lieu, nous abordons la problématique de la spécification et le déploiement des politiques de sécurité hétérogènes (par exemple, les politiques de confidentialité, de protection de la vie privée, d'intégrité, etc.) sur les données externalisées. À cet effet, nous avons, tout d'abord, défini un langage formel suffisamment expressif permettant de spécifier le plus finement possible les besoins hétérogènes de sécurité et de fonctionnalité sur les données externalisées, ainsi que les mécanismes de sécurité existants pouvant être utilisés pour satisfaire et déployer ces besoins. Ensuite, nous définissons une méthode de raisonnement pour notre modèle formel permettant de choisir automatiquement la combinaison de mécanismes de sécurité assurant un compromis optimal entre la sécurité, la fonctionnalité des données externalisées, et la complexité de son déploiement sur le système utilisé. Enfin, nous utilisons une méthode basée sur le teintage de données afin d'améliorer notre méthode de raisonnement précédemment définie.

Acknowledgement

I would like to sincerely thank all people how have contributed in one way or another to the accomplishment of this work. Some of them come circumstantially to encourage us or just to listen to us when we need someone to talk with, or when we fail to find answers to our multiple questions.

First of all, I would like to thank from very deep inside my advisors, **Frédŕic Cuppens**, **Nora Cuppens-Boulahia**, and **David Gross-Amblard**, for their support, their dedication, their trust and their advices throughout the three years of my thesis. It has been an honor for me to be one of their Ph.D. students. I would like to express my gratitude to **Nora** for her precious advices, for her answers to all my questions (even to meaningless ones), and for her optimism that helped me in reaching this important goal.

I would also like to thank **Règine Laleau** and **Michaël Rusinowitch** who had the hard task of reporting my thesis and giving their advice to improve its content. Thanks a lot to , **Refik Molva**, **Pascal Lafourcade**, and **Louis Rilling** for being part of the jury of my thesis.

I would like to thank my family. All the pages of this thesis would not be enough to express my gratitude to them: their encouragement, their teaching, and their love have been, and will always be, a principal reference point for me. I am deeply indebted to my parents for their endless love, support and encouragement. All my thankfulness is for: my two sisters and my brother, who never left my side and are very special to me. My parents, sister and brothers in law, their support and affection encourage me to attend my objective.

Most importantly, I would like to thank, my truly God's gift, lovely, and sweet wife, who has been my inspiration. I could never have accomplished this thesis without her love, support, and understanding. She makes my life a joy.

Finally, A huge thank to all my friends and colleagues, members of our SERES team, **Tarik**, **Tarek**, **Reda**, **Said**, **Safaa**, **Lyes**, **Nada**, **Fabien**, **Patrick**, **Eric**, **Samiha**, **Khaoula**, **Stéphane** and **Yanhuang**. A especial thank to **Sabir** and **Mariem**, who always find a way to disturb me, but who supported me so much. They have all been my family in France, thanks for being a part of this adventure.

Contents

CHAPTER 1

Introduction

The ever-increasing of technological advancements is breaking down the classical way of electronic data storage and retrieval. Principally for economical benefits, these days, both individuals and companies are increasingly using remote storage services (e.g., Google Drive [GoogleDrive], Dropbox [Dropbox] and CloudMe [CloudMe]). These storage services enable data sharing and ensure availability of data from anywhere at any time. However, this new paradigm brings several cloud-specific security issues, particularly when the storage servers offering such services are untrusted.

1.1 Data Outsourcing Basics

In the last decade, the popularity of cloud storage services has increased dramatically because of the explosive growth of digital contents. According to the US International Data Corporation, the digital universe will increase by a factor of 300 to reach around 40 trillion gigabytes of outsourced data by 2020 [J. Gantz 2012]. This rapid growth of digital universe is raising the need for new storage space. For these reasons, private organizations and companies need to make large investments into their IT infrastructure. Additional hardware and software are required, as well as staff for its operation and maintenance. However, these expenses are contradictory with the perpetual need to reduce costs in order to stay competitive. As a consequence, Cloud storage-based services are being more and more attractive since they are providing user-friendly, easily accessible and cost-saving ways of storing arbitrary data in a pay per use business model. Business organizations and government agencies does no longer need to spend large amounts of money on buying and managing complex software and hardware systems that will be used to store collected data. In several ways, Cloud storage is collapsing our models of what is accepted as being possible and even reasonable to do with computers.

Cloud storage is further distinguished between public, private, community and hybrid Clouds. A private cloud is hosted internally and operated by the end user and is used inside its intranet only or by using VPN access from outside. In contrast to a private cloud, a public Cloud is located externally to its end users and is open and accessible for every one since it can be used by both business users and private users. Community Clouds allow several independent entities to profit the cost benefits of a shared nonpublic cloud. Community Clouds have huge potential for companies and entities having to comply with identical regulatory, or legal restrictions. Finally, hybrid Clouds are just as the name indicates, are a combination of two or more distinct cloud infrastructures (e.g. public and private Clouds).

According to their deployment models as public, private or hybrid clouds, cloud systems are classified to support three cloud service models [Mell and Grance 2011]. Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

- *Software as a Service (SaaS)*: A SaaS cloud implementation provides generally software or, in more general way, an application to its end users. There is no need for end users to be concerned with the backend infrastructure used by the application. The backend infrastructure of the provided application are hidden and offered *as-a-service* behind the scene of that application. The increasing number of applications that are provided, such as Google's GMAIL [GMail], Yahoo Mail [YahooMail] or Microsoft Share Point [Microsoft], illustrates the prosperity of SaaS.

- *Platform as a Service (PaaS)*: A PaaS cloud provides a package of software and infrastructure as a programmable environment (called a *container*). With PaaS, the provided service is represented by the entire application environment including the computing platform as well as the development and solution stack. This cloud service model is mostly used by new startups and small companies since it allows them to develop and deploy their solutions and services without the need to acquire in-house servers and working teams to manage them. Salesforce.com's Force.com platform [Salesforce] and Google Google App Engine [Google] are excellent examples of a PaaS architecture.

- *Infrastructure as a Service (IaaS)*: Generally speaking, an IaaS cloud provides virtualized resources, such as virtual machines that are fully controlled by costumers, including choosing operating system and the amount of needed storage space that best meet their requirements. However, IaaS cloud users cannot manage or control the underlying cloud infrastructure. Amazon's Web Services [Amazon]

and RackSpace's Cloud Services [RackSpace] are both good examples of IaaS providers.

1.2 Data Outsourcing Security Challenges

Data outsourcing consists in deporting on remote servers storage and management of information traditionally stored and managed by in-house facilities, which basically signifies entrusting data to a third party with which no prior trust relationship has been established. In the first hand, individuals who may use Cloud storage services to store their personal information want to be sure that only specific persons can access it, and of course this should also exclude storage service providers, since there is no tangible reasons for them to access the outsourced data. In the other hand, the data collected and managed by companies contain often highly sensitive information which poses a serious threat to a company's business when they are disclosed to unauthorized parties.

Many incidents (e.g., [Mulazzani et al. 2011], [Newton 2011] and [Lewis 2014]), in which Cloud storage services' vulnerabilities have been exploited, prove that doubts about their usage are well justified. According to a recent study led by Verizon [Verizon 2015], around 80,000 security incidents have been detected in 2014 and that around 2200 of these security incidents have caused a leakage of information.

In fact, data outsourcing paradigm gave birth to several security issues particularly when the data is stored and managed by a honest-but-curious server, These security issues are related mainly to four problems as briefly outlined in the following.

- **Data Confidentiality.** The data collected and managed by companies and organizations contain often highly sensitive information that should not be disclosed to unauthorized entities including cloud storage service providers. The US National Institute of Standards and Technology (NIST) [NIST 1985] specifies that the Cloud users' confidential data is disclosed to an unauthorized entity if it has the privilege to access the outsourced data, to collect the users' confidential data, and to understand the meaning of the collected confidential data. Indeed, sensitive outsourced data confidentiality remains one of the greatest security issues with regards to cloud computing, as the data will be controlled and managed by potentially untrustworthy cloud service storage providers, since according to [datalossdb 2015], around 40% of data breach incidents are caused by cloud services' insiders. It is then challenging to define new solutions that grant data

confidentiality while allowing an efficient manipulation (e.g., search, modification, and computation) of the outsourced data.

- **Data Privacy.** The data stored and processed may be a subject to regulatory and compliance requirements. Recent European regulation [UE2 2014] explicitly specifies that specific categories of identifier information should be either encrypted or kept separated in order to grant privacy. Therefore, it is interesting for data owners or data collectors to implement additional security controls that meet regulatory or legal requirements even when an underlying Cloud storage service provider that does not fully meet those same requirements is used to outsource the data.

- **Data Integrity.** Integrity is yet another critical concern with regards to cloud environments. It concerns both data storage and data process integrity. Data storage integrity means that data have to be stored honestly by Cloud service providers and any integrity violation (e.g., unauthorized modification or loss) should be detected by the Cloud user who has outsourced the data. Querying and computation integrity means verifying whether or not queries and computations are faithfully performed by the Cloud storage service provider over the outsourced data. Hence, storage and process integrity need to be taken in consideration when designing a system for ensuring security of data stored and managed by a honest-but-curious Cloud storage service provider.

- **Data Ownership.** Further outsourced data concern emerges with the ownership of information assets. Data ownership and being responsible as a data custodian are fundamentally different. There is potential for erosion of information asset ownership when moving valuable data to any external Cloud storage service provider. For illustration purpose, if we suppose that a valuable outsourced data has been used by an untrustworthy Cloud storage service provider or by an external adversary to make economical profits, it is then interesting for the data owner to be able to accuse the entity having illegally used the outsourced data.

1.3 Objectives and Contributions

Data outsourcing rises many challenging security issues, mainly due to the loss of physical control. These challenges influence significantly on the security of the outsourced data.

On one side, enforcing data confidentiality in Cloud storage environments becomes more challenging when the data is stored and managed by untrusted third party. One

possible solution consists of encrypting the data to be outsourced on the client machine (which is supposed to be trusted) before uploading it to the Cloud storage server. Encrypting the outsourced data is considered as the last effective line of defense allowing to protect the outsourced data from both external hackers and malicious Cloud storage server administrators, since obviously, if the encryption keys are not compromised by a hacker or a malicious administrator that manages the server, the confidentiality of the outsourced data remains ensured. This solution is useless when dealing with big in-production databases since it raises significantly the cost of data querying, as in order to process queries, we need to read back the encrypted data from the server to the client, decrypting the data, and executing the queries on the client machine. The first objective of this thesis is to define new solutions allowing to ensure the best compromise between outsourced data confidentiality and utility. To meet this objective, we propose the following contributions:

- *Contribution 1.* We propose an approach [Bkakria et al. 2013a, Bkakria et al. 2013b] allowing the protection of confidentiality of sensitive information in outsourced multi-relational databases by improving existing approaches [Ciriani et al. 2007, Ciriani et al. 2009] based on a combination of data fragmentation and encryption. These approaches have a major limitation as they assume that data to be outsourced is represented within a single relation schema (or table) which is too strong and seldom satisfied in real environments. Then we define a secure and effective technique for querying data hosted on several service providers. Finally, we improve the security of the querying technique in order to protect data confidentiality under a collaborative Cloud storage service providers model.

- *Contribution 2.* We define a policy-based configuration framework [Bkakria et al. 2014b] for encrypted data allowing the data owner to specify the policy to be applied over the outsourced data. Then, we provide an efficient method allowing to detect conflicts between confidentiality requirements (e.g., the set of sensitive information) and utility requirements (e.g., SQL queries that should be executed over the encrypted data) specified by the data owner. Finally, we propose an heuristic polynomial-time algorithm for finding a combination of encryption schemes that satisfies a near optimal trade-off between confidentiality requirements and utility requirements.

On the other side, by analyzing some real-life scenarios of applications that need mechanisms to securely outsource data, we realize that the security and utility requirements specified by data owner are different in each scenario. In addition, they are

in some cases heterogeneous (e.g., confidentiality requirements, privacy requirements, ownership requirements, etc.). Security mechanisms allowing to enforce those security requirements have recently been the focus of huge interest, especially cryptographic and information hiding techniques. These techniques greatly help in tackling security issues: copyright protection (cryptography, watermaking, fingerprinting), content/data confidentiality (cryptography through encryption, fragmentation, access control), content/data integrity (cryptography through digital signature or message authentication codes, watermaking), authentication of entities (cryptography), anonymity (anonymous networks or granting), and privacy (cryptography, k-anonymity and its extensions, and the more recent differential privacy). These mechanisms are known to be efficient when used independently. However, in many situations they have to be combined in an appropriate way to provide the security functionalities without one harming the other. The second objective of this thesis is then to design support tools that allows data owners to easily specify their security requirements and automatically choose the best set of security mechanisms, and the best way to combine them (e.g. the best order in which they are applied) to get the best tradeoff between complexity, security and utility in the final choices. To meet this objective, we propose the following contributions:

- *Contribution 3.* Using an Epistemic Linear Temporal Logic (Epistemic LTL), we defined an expressive language [Bkakria et al. 2014a] allowing to: (1) formally model a system composed of involved entities (e.g., data owner, Cloud Storage server administrator, external adversary, etc.) and the data structure on which the security policy should be enforced. (2) formally express as finely as possible the security policy defined by the data owner. Then, we define a reasoning method for our formal model allowing to identify the relevant combination of mechanisms to efficiently enforce the defined security policy.

- *Contribution 4.* In [Bkakria et al. 2014a], we suppose that the security mechanisms that can satisfy a policy are applied parallelly over the target system. However, we have seen that in some cases, some security mechanisms should be applied over the same part of the data to be outsourced to satisfy the required security properties. Obviously, in those cases, we should take into consideration conflicts that may occur between security mechanisms which makes finding a combination of security mechanisms that satisfy many security requirements much harder to fulfill. We define an approach that extends [Bkakria et al. 2014a] and uses a planning graph based method to find the combination of security mechanisms providing the near optimal trade-off between the security and the utility of the data to be outsourced and the complexity of its application over the used system.

- *Contribution 5.* Using our reasoning method proposed in the previous contribution, defined policies over outsourced data are either wholly satisfied or violated. This contribution strive to overcome this limitation by designing an approach allowing outsourced data owner, in the case in which no combination of security mechanisms could fully satisfy the chosen policy, to come up with the best compromise between security constraints to be enforced and the set of goals to satisfy over the outsourced data. To this end, we extend the planning graph based approach presented in [Kautz and Selman 1999] by using a data tainting based method allowing to (1) mark the set of fact nodes that violate safety constraints and (2) effectively propagate those taints to fact nodes representing the goals that need to be satisfied. Later on, based on the propagated taints, we define a reasoning method allowing to get the near optimal compromise between the goals to satisfy and the security constraints to ensure.

1.4 Organization of the dissertation

This dissertation is divided into 2 parts described below.

Part I – Preserving Outsourced Data Confidentiality – this part focuses on data confidentiality preservation which becomes more challenging when the data is outsourced to an untrusted Cloud storage provider.

Chapter 2 – Outsourced Data Confidentiality: State of the Art – we discuss research investigations and technologies aiming to overcome cloud data confidentiality issue. It depicts the main results published in the data outsourcing scenario, focusing on data confidentiality and mechanisms for outsourced data querying and processing.

Chapter 3 – Preserving Multi-relational Outsourced Databases Confidentiality using Fragmentation and Encryption – proposes to combine data fragmentation and encryption to ensure the confidentiality of multi-relational Databases stored at a honest-but-curious cloud server and provides an efficient and secure technique for querying outsourced data.

Chapter 4 – Combining Encryption-based Mechanisms to ensure Outsourced Data Confidentiality – proposes a policy-based configuration framework that combines encryption-based security mechanisms over outsourced data to ensure the best trade-off between the confidentiality and the utility of the outsourced information.

Part II – Heterogeneous Security and Utility Requirements Specification and Enforcement over Outsourced Data – this part focuses on defining approaches allowing to specify and enforce heterogeneous security and utility requirements (e.g., confidentiality, privacy, ownership, computation, etc) over outsourced data.

Chapter 5 – A new Epistemic Temporal Logic based language for Specifying Security Policy and Security Mechanisms – proposes a formal model allowing the specification of heterogeneous security and utility requirements over a data structure to be outsourced, as well as the specification of existing security mechanisms that can be used to enforce them.

Chapter 6 – Formal Reasoning Method to enforce Security Policies over Outsourced Data – defines a reasoning method for our formal model allowing outsourced data owners to automatically get the combination of security mechanisms providing a near optimal trade-off between **the security and the utility** of the data to be outsourced and also **the complexity** of its application over the used system.

Chapter 7 – Best effort based approach for security mechanisms planning to enforce security policies over outsourced data – extends the reasoning method proposed in Chapter 6 and proposes an approach that gets the mechanisms execution plan that provides the best compromise between **security constraints** to enforce and the set of **goals** to satisfy over the outsourced data.

Chapter 8 – Conclusions and Perspectives – this Chapter concludes the dissertation by summarizing the contributions and presenting the perspectives for future work.

Part I

Preserving Outsourced Data Confidentiality

CHAPTER 2

Outsourced Data Confidentiality: State of the Art

2.1 Introduction

Cloud storage is an emerging paradigm, outsourcing the computation and storage capabilities to external service providers. Mainly because of this loss of control on outsourced data, data owners hesitate using Cloud storage services. During the last few years, outsourced data confidentiality concerns was becoming more and more legitimate regarding the latest data breach and capture mediated events. In November 2013, the Washington Post reveals more indiscriminate data capture, by the US National Security Agency (NSA). This collection of data is done by intercepting private communications that links Google and Yahoo data centers around the world and decrypting the traffic that should be protected in transit [1] [NSA 2013b]. As a result, in December 2013, a survey conducted by PriceWaterhouseCoopers points that around 50 percents of companies in Germany consider storing data in the Cloud risky after hearing about NSA data spying [NSA 2013a]. To cope with this confidentiality problem, several approaches have been proposed.

In order to protect the confidentiality of outsourced data in an *honest but curious* server storage model, outsourced sensitive information are encrypted preventing the access of outside unauthorized entities (attackers) as well as insider entities (malicious administrators) from the Cloud server itself [Davida et al. 1981]. This solution introduces an interesting research challenge consisting of the problem of efficient outsourced encrypted data querying. When dealing with encrypted data, confidentiality requires that data decryption operations should be only performed at the data owner

[1] Data in transit is data being accessed over the network, and therefore could be intercepted by someone else on the network or with access to the physical media the network uses

site. Therefore, enabling external entities to directly execute queries on encrypted outsourced data became a key challenge. In addition to encryption-based approaches, secret sharing based approaches indicates a promising prospect to solve the challenges of preserving outsourced data confidentiality. Moreover, in specific domains, outsourced data confidentiality can be ensured using other approaches, such as data fragmentation. These methods suppose that the associations between different data objects to be outsourced are more sensitive than their values.

In this chapter, we survey the main existing approaches addressing the confidentiality issues arising from the loss of control on outsourced data. The remainder of the chapter is organized as follows. Section 2.2 gives an overview of the main existing encryption-based approaches and their proposed methods for querying encrypted data. Section 2.3 presents and discusses existing fragmentation-based approaches to ensure outsourced data confidentiality. Finally, we conclude this chapter in Section 2.4.

2.2 Outsourced data Confidentiality by Encryption

Despite that traditional encryption mechanisms (e.g., Data Encryption Standard (DES) [NIST 1999] and Advanced Encryption Standard (AES) [FIPS. 2001]) ensures strong confidentiality guarantees. They are suffering from two main limitations that reduce the effectiveness of these traditional schemes, especially because of the large amount of outsourced data.

First, when outsourcing the data to an untrusted (*honest by curious*) service provider, data owners generally encrypt the data before its outsourcing to the remote storage server. However, the employment of traditional encryption schemes is inconvenient for huge amount of data as it reduces significantly the computation capacity both at the client and the server sides.

Second, these traditional encryption approaches are deterministic, they are not adaptable and do not allow performing operations over encrypted data (e.g., search, computation, etc.).

Search over encrypted data is very useful to overcome the traditional all-or-nothing retrieval policy of encrypted data. When designing an approach for searching over encrypted data, the computation on the client-side, as well as the communication overhead, plays critical roles for deciding its efficiency. More the computation on the client-side is minimized and the communication overhead is reduced, more the searching over encrypted data approach is efficient.

Several approaches have emerged to allow efficient search over encrypted outsourced data. They can be classified into several classes: Bucketing-based approaches [Hacigümüs et al. 2002, Hore et al. 2004], techniques that are based on some specialized data structures [Damiani et al. 2003b, Damiani et al. 2003a, Shi et al. 2007, Park 2011, Wang et al. 2014], order preserving encryption based approaches [Agrawal et al. 2004, Boldyreva et al. 2009, Boldyreva et al. 2011, Boldyreva et al. 2012], Searchable Encryption based approaches [Song et al. 2000, Goh 2003a, Golle et al. 2004, Curtmola et al. 2006, Amanatidis et al. 2007, Chase and Kamara 2011, Kamara et al. 2012, Kamara and Papamanthou 2013], and Homomorphic Encryption based approaches [Boneh et al. 2005, Paillier 1999, Gentry 2009].

2.2.1 Bucketing Based Approaches

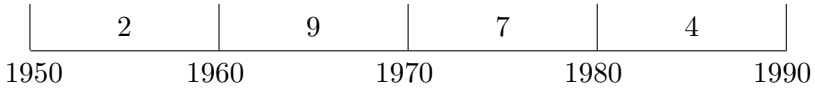
Hakan Hacigümüs et al. [Hacigümüs et al. 2002] have introduced a model for ensuring the confidentiality of outsourced databases based on a bucketization-based technique. This technique consists of segmenting the domain of attributes, which their assumed values are considered sensitive, into a number of non-intersecting subsets of values called buckets and having the same size. Then, a unique random identifier ID is associated to the set of unencrypted values in each Bucket. Finally, the encrypted tuples together with buckets IDs are outsourced to a cloud storage server. Let us suppose that a database $\mathcal{D}$ composed of a finite set of relational tables $\{T_1, \cdots, T_n\}$ has to be outsourced. For each tuple $t_i = \{v_1^i, \cdots, v_{m_i}^i\}$ in the relational table T_i of $\mathcal{D}$, $t_i^s = \{etuple, ID_{v_1^i}, \cdots, ID_{v_{m_i}^i}\}$ will be outsourced to the cloud server, where *etuple* represents the encryption of the tuple $(v_1^i, \cdots, v_{m_i}^i)$ and ID_j^i represents the identifier of the bucket containing each v_j^i, $1 \leq j \leq m_i$. The bucketization method (segmentation function) used to generate the corresponding bucket IDs for the values of each attribute are stored privately by the data owner.

For illustrative purpose, let us consider that the relational table *Patient* (Figure 2.1 (a)) will be outsourced. Figure 2.1 (b) represents the bucketization method used for the values of the attribute *Yob* and Figure 2.1 (c) represents the encrypted form of the relation *Patient* that will be outsourced. The values of the attribute I_C in Figure 2.1 (c) are indexes obtained by the application of the bucketization method in Figure 2.1 (b) for the attribute Yob. In order to query the outsourced relation, an authorized user replaces the values used to search over each attribute by their corresponding buckets IDs calculated using the adopted bucketization method. For example, if an authorized user wants to execute the query q : $\pi_{Patient.SSN}(\sigma_{Patient.Dob=1965}(Patient))$ over the

(a)

SSN	Name_pat	Yob	Illness
865746129	A. Barrett	1976	Illness_1
591674603	C. Beat	1981	Illness_2
880951264	N. Baines	1986	Illness_1
357951648	S. Brandt	1951	Illness_3

(b)



(c)

etuple	I_A	I_B	I_C	I_D
Njk6%lk$djI=w	19	101	7	52
*lNvsdIkj**/w=u	31	18	4	42
ppUt*%diHgW	77	18	4	96
JgedŶdd#=Az	41	83	2	0

Figure 2.1 – (a) *Patient* relation, (b) partition function used for attribute Dob, and (c) the encrypted relation $Patient^s$ stored in the cloud server.

outsourced relation, he or she needs to rewrite the query q by replacing the value 1965 by it corresponding bucket ID 9.

Hacigümüs's bucketization technique suffers from three main disadvantages. First, despite this method allows to perform exact search over outsourced relational databases as equal values are indexed with equal bucket IDs, it does not support order search since the bucketization method used to generate bucket IDs does not necessarily preserve the plaintext domain ordering (e.g., the bucketization method used in Figure 2.1 (b)). Second, it is clear that the final results provided by this encrypted data querying method contain false positives and needs to be filtered to remove out any tuples that do not satisfy the original query conditions. Third, this technique may give rise to a possible privacy violations at server side.

Enhancing Hacigümüs's technique, Hore et al. [Hore et al. 2004] investigated firstly the problem of performing order search over encrypted relational databases while minimizing the number of false positives in the queries results. In a second time, they studied the leakage of information due to the bucketization-based technique. A two levels analysis is provided. In the first level, the leakage of information due to the publication of the bucket IDs of only one attribute is considered. In the second level, they consider the leakage of information due to the publication of the bucket IDs of all at-

tributes. They propose two metrics for privacy measures relevant to data bucketization: a measure based on the distribution of the values within each bucket and a measure based on the distribution entropy of the values within a bucket. To ensure the best level of privacy, the data owner has to maximize the distribution entropy of the values within each bucket. Finally, they prove that finding the best trade-off between data privacy and querying efficiency when using bucketization techniques is NP-complete. To overcome this problem, they propose an heuristic-based method fixing a maximum permitted degradation threshold of querying performance.

Another problem related to this bucketization techniques is that, to achieve the best level of data protection, the buckets should all have the same size (the number of tuples in each bucket is the same). Despite this property can be ensured in the case of data at rest databases [2], it is too hard to maintain it for data in motion databases [3].

2.2.2 Specialized Data Structures Based Techniques

Damiani et al. [Damiani et al. 2003a, Damiani et al. 2003b] introduced an approach allowing to perform exact and order searches over encrypted data using a B^+– tree based indexing method. The proposed technique is used for each attribute to efficiently allows order and exact searches over it. That is, a B^+– tree is created over the unencrypted values of the attributes to be used to search the data. Then, for each node in the created trees, a couple $(id, enode)$ is created and outsourced to the cloud server, where id is a node identifier and $enode = \langle l_{id}, E(value), r_{id} \rangle$ represents the encrypted form of the cleartext node using a deterministic encryption E, l_{id} and r_{id} represent respectively the left and the right nodes identifiers.

To perform an order search (i,e,. range query) over an attribute $attr$, as a first step, the client (or data owner) selects the root node of the outsourced B^+– tree created over the values of $attr$. In the second step, the client decrypts $E(value)$ and compares $value$ with the search condition to figure out with branch (left or right) is to be taken. This procedure repeats until the leaf level of the B^+– tree is reached. Finally, the client navigates through the leaves to select the corresponding values. Although the final querying results produced using this method are accurate (no false positive), to perform a single search operation, we still need to perform many interactions (the height of B^+– tree) between the client and the cloud server.

Shi et al. [Shi et al. 2007] proposed MRQED – a confidentiality-preserving searchable scheme supporting multi-dimensional order search over encrypted data. In this

[2] inactive data which is stored in relational database

[3] active data which is stored in relational database

approach, a discrete integer values 1 throught D is used to encode each attribute, where D represents the number of possible values that might be assumed by each attribute. Therefore, a binary interval tree $BIT(D)$ is created over integers 1 from D. Despite this approach ensures provably strong security, it is useless in practice since a server needs to fully scan the whole encrypted relation to perform single order search request.

Based on the work proposed by Dan Boneh et al. in [Boneh and Waters 2007], a public key based approach called Hiden Vector Encryption (HVE) is introduced in [Park 2011] supporting equality and order searches over encrypted data. Nonetheless, the complexity of range search per data item is linear in the range size, which can be too much expensive in terms of execution time when the range size is large. Moreover, the proposed technique does not use any form of indexing to reduce access complexity that might be extremely expensive when dealing with large datasets.

Recently, Wang et al. [Wang et al. 2014] enhance the efficiency of the approach proposed in [Shi et al. 2007] by introducing Maple – a scalable multi-dimensional range search over encrypted outsourced data with MRQED tree-based index. They formally define the leakage function and security game associated to a tree-based public-key MDRSE. Then, by combining R-Trees [Guttman 1984] and HVE [Boneh and Waters 2007], they improve search efficiency while the protection of single-dimensional range queries' privacy is ensured.

2.2.3 Order-Preserving Encryption Based Approaches

An order-preserving encryption (or OPE) scheme is a deterministic symmetric encryption scheme based on an encryption algorithm that produces ciphertexts preserving the order of the plaintexts. Formally speaking, let $\mathcal{D}$ and $\mathcal{D}^e$ be finite ordered sets (for the sake of simplicity, we can consider them to be subsets of natural numbers). OPE is an order-preserving encryption with plaintext space $\mathcal{D}$, ciphertext space $\mathcal{D}^e$, and key space $\mathcal{K}$ if and only if for any key $k \in \mathcal{K}$ and any plaintext values $v_1, v_2 \in \mathcal{D}$, $v_1 < v_2$ then $OPE(v_1, k) < OPE(v_2, k)$.

OPE has been introduced to the database community by Agrawal et al. [Agrawal et al. 2004] as an approach to efficiently perform order search over encrypted databases. The OPE algorithm is constructed in three steps: (1) model the input and target distributions, (2) flatten the plaintext data into a flat database, and (3) transform the flat database into the cipher database. It allows a remote untrusted database server to index the data it receives in encrypted form, in a data structure that permits time logarithmic range queries (in the size of the database). However, this approach

can only deal with numerical data and gives rise to serious security risks associated with revealing data order.

Through a formal cryptographic study of OPE, Boldyreva et al [Boldyreva et al. 2009] prove that order-preserving technique proposed by Agrawal et al. [Agrawal et al. 2004] does not fulfill all the standard notions of security since the used encryption algorithm is required to be deterministic, which discloses the frequency of each distinct value in the database. OPE security is measured by comparing it with an ideal object satisfying the order-preservation and information hiding properties. This object is defined by Boldyreva et al. to be a random order-preserving function (ROPF) which is randomly chosen from the set of all order-preserving functions that map $\mathcal{D}$ to $\mathcal{D}^e$. Although the well-understood notion of random function, ROPF is a less intuitive object since it is unclear what information ROPF leaks about the encrypted data. Boldyreva et al. characterize the security offered by the ideal ROPF function and warn that the provided characterization is a weak.

One-wayness is a cryptographic property requiring that given a function, evaluated using a random value, an adversary is enable to invert it. It is one of the most basic (and weakest) cryptographic properties and was not ensured for ROPF proposed in [Boldyreva et al. 2009]. The *one-wayness* property is shown to hold in a subsequent papers [Boldyreva et al. 2011, Boldyreva et al. 2012]. These papers introduce what they call a necessary adjustments to the typical definition of *one-wayness*. In the new definition, an adversary cannot forward-evaluate ROPF, since otherwise the knowledge of the secret key is required. Nonetheless, this is a relatively hard requirement, which is seldom satisfied in practice. Moreover, their new definition uses what they call the *uniformity assumption* requiring the pre-image to be uniformly random distributed. The authors notify in advance that the provided security analysis of ROPF may not hold in cases where the *uniformity assumption* is not ensured. In addition, they indicate (and we fully agree) that in practice, the *uniformity assumption* is extremely difficult to ensure.

Boldyreva's OPE is proved to be one-way in the previously described restricted sense. Nonetheless, Kolesnikov and Shikfa [Kolesnikov and Shikfa 2012] prove that standard stronger security notions do not provably hold for ROPF (and Boldyreva's OPE). Finally, it was concluded in [Xiao and Yen 2012] that order-preserving encryption leaks at least half of the plaintext bits. As a conclusion, we said that despite OPE based approaches are not proved to ensure standard stronger security notions, they are better than no encryption at all. We believe that this kind of encryption should only be used in the cases where the leakage of some information about the data to be outsourced will not cause grave damage to the data owner.

2.2.4 Searchable Symmetric Encryption Based Approaches

The first practical scheme for searching over encrypted data (SWP) was proposed by [Song et al. 2000] based on the use of a special two-layered encryption construct allowing searching the encrypted data with a sequential scan. The idea consists of encrypting each word separately and then a hash value having a special format is embedded inside the ciphertext. This encryption method gives the server the search ability by extracting this hash value and checking if the value is of a given special form (which indicates a match). The main disadvantage of (SWP) is that their specific two-layer encryption method has to be used, which makes it useless, for example, when dealing with compressed data. From the security point of view, SWP leaks the potential positions of the queried keywords in a document. As a consequence, by performing several queries, it is possible for an adversary to use statistical analysis to figure out the words inside the documents.

In [Goh 2003a], Goh attempts to overcome some of the limitations of the SWP scheme (e.g, special document encryption and fixed-size words) using a Bloom filter (BF) [Bloom 1970] as a per-document index. That is, a new index is embedded for each encrypted file. This index is independent of the underlying encryption algorithm. In one hand, the use of a BF per document reduces the search time to linear in the number of documents. In the other hand, Bloom filters introduce two serious problems: (1) an accuracy problem; That is, the possibility of false positives which can be reduced to an acceptable level by using appropriate parameter settings. (2) a security problem; That is, in each BF related to a document, the number of 1s is a function of the number of BF entries which represents in reality the number of distinct keywords per document. As a result, Goh' scheme leaks the number of keywords in each document.

Golle et al. [Golle et al. 2004] propose the first conjunctive keyword search scheme based on the idea assuming that special keyword fields are associated with each document. For instance, in the case of emails, the keyword fields might be *From*, *To*, and *Subject*. Conjunctive keyword search means that a user is able to find documents containing all of several keywords in a single search query. However, with the proposed scheme and in order to search over the encrypted documents, the user is required to know in which keyword field the search should be performed. Unfortunately, this scheme does not scale for large databases since the costs of communication and storage linearly depend on the number of stored documents in the outsourced database. The security of Golle et al.'s scheme relies on the Decisional Diffie-Hellman (DDH) assumption [Boneh 1998].

Curtmola et al. [Curtmola et al. 2006] proposed two new schemes based on the idea of adding an inverted index. Instead of an index per document, an inverted index create an index per distinct word in the database, which reduces the search time to sub-linearity to the number of documents containing the keyword. This is not only sub-linear but also optimal. Both proposed schemes are based on the use of FKS dictionary [Fredman et al. 1984] as a look-up table which compacts more the used index and decreases the look-up time to O(1). Unfortunately, updating encrypted data is expensive because of the way in which the data is stored in the server. In consequence, the proposed schemes are more suitable for data in rest than dynamic data.

Amanatidis et al. [Amanatidis et al. 2007] proposed two new constructions based on the use of deterministic message authentication codes (MACs) to perform search over encrypted data. The first proposed scheme (Curt-I) relies on appending a deterministic MAC to a secure encryption of a keyword indistinguishable against chosen plaintext attacks (IND-CPA). Informally speaking, an encryption scheme is IND-CPA secure if an adversary $\mathcal{A}$ cannot distinguish the encryption values of two arbitrary messages chosen by $\mathcal{A}$, even if $\mathcal{A}$ can adaptively query an encryption oracle. In the second proposed construction (Curt-II), the MAC of the plaintext is used as the randomness inside of the encryption. Curt-I allows a client to search by simply generates the MAC of a keyword and stores it together with the encrypted keyword on the server. Then, the server will use the indexed MACs to find the correct answer. With Curt-II, the user calculates and embeds the MAC inside the ciphertext of the keyword allowing the server to search for the queried ciphertexts. Curt-I is proved to be secure if and only if the encryption scheme is IND-CPA secure and the MAC is unforgeable against chosen message attacks. Curt-II is proved to be secure under the assumptions that the encryption scheme is IND-CPA secure and the MAC is a pseudo-random function.

Based on the schemes proposed in [Curtmola et al. 2006], Chase and Kamara [Chase and Kamara 2011] proposed an adaptively secure construction relying on the generation of an inverted index to create a padded and permuted dictionary. An optimal search time can be provided through a hash tables based implementation of the dictionary. Conceptually, the proposed scheme is IND2-CKA [Goh 2003b] secure hiding the data structure. However, the proposed construction still disclose the access and search pattern.

Kamara et al. [Kamara et al. 2012] extend the constructions proposed in [Curtmola et al. 2006] to allow efficient updates (add, delete, and modify documents) over the encrypted data. The proposed extension is based on adding special arrays called "deletion arrays" to keep track of the search array positions that need to be

modified in case of an update query. Moreover, in order to modify the pointers without decrypting them, the proposed scheme uses homomorphically encrypted array pointers. From the security point of view, update operations in the defined scheme leak the trapdoors of the keywords contained in an updated document.

Thanks to the advances in multicore architectures, Kamara and Papamanthou [Kamara and Papamanthou 2013] proposed a highly parallelizable new scheme that provides a new way to achieve sublinear search time. It is based on the use of data structure called keyword red-black (KRB) trees which are similar to binary trees having pointers to a file as leaves. Each node in a KRB tree stores information when at least one of its following nodes is a path to a file identifier containing the keyword. Conceptually, the security definition of the proposed scheme is a generalization of IND2-CKA. The authors proved the security of their construction under the random oracle (RO) model.

2.2.5 Homomorphic Encryption Based Approaches

Homomorphic encryption are based on cryptographic schemes whose encryption function is a homomorphism. That is, they preserve group operations performed on encrypted data. Homomorphic encryption algorithms give the ability to a third party to perform computations over encrypted data which ensures privacy preservation.

Rivest et al. [Rivest et al. 1978] introduced the idea of performing soft computations on encrypted data. Their motivation was the ability to use an untrusted third party to store an encrypted database and allows the owner to perform simple updates and queries while ensuring that nothing about the database contents is revealed.

Homomorphic cryptosystems are mainly defined over algebraic groups or rings [Cohn 2000]. In the first hand, algebraic groups based homomorphic cryptosystems allow a single operation to be performed over encrypted data, usually denoted by multiplication or addition.

Definition 1. ***(Algebraic group-based homomorphic encryption.)*** *An encryption scheme $\mathcal{S} = (Enc, Dec, \mathcal{K})$ is homomorphic if for all $k \in \mathcal{K}$, it is possible to define groups $\mathcal{M}$ and $\mathcal{C}$ so that:*

- *$\forall m \in \mathcal{M} : c = Enc(m)$, it holds that $c \in \mathcal{C}$.*
- *$\forall m_1, m_2 \in \mathcal{M}, \forall c_1, c_2 \in \mathcal{C}$ with $m_1 = Dec(c_1)$ and $m_2 = Dec(c_2)$, it holds that:*

$$Dec(c_1 \circledast c_2) = m_1 \circledast m_2$$

where $\circledast$ are the respective group operations in $\mathcal{C}$ and $\mathcal{M}$.

In the literature, algebraic group based homomorphic cryptosystems are considered as partially homomorphic encryption (PHE). Several PHE cryptographic systems were proposed to allow simple computations over encrypted data. El Gamal [ElGamal 1985] proposes a cryptosystem that computes multiplication over encrypted data. Goldwasser and Micali [Goldwasser and Micali 1982] propose a cryptosystem that computes XOR of encrypted bits. Paillier [Paillier 1999] proposed a cryptosystem that is able to computes addition over outsourced data. Later on, Boneh et al. [Boneh et al. 2005] introduce the BGN cryptosystem allowing to perform an arbitrary number of additions, one multiplication, followed by an arbitrary number of additions. PHE provides *semantic security* [Goldreich 2004] which represents a strong security guarantees. Informally speaking, semantic security means that any adversary knowing only the public key and an encrypted value cannot learn any information about the underlying unencrypted value, other than its length. Due to their specialized nature, PHEs are quite efficient and can be used in practice.

In the other hand, ring-based homomorphic cryptosystems naturally support two operations: addition and multiplication. Definition 1 can be extended to a ring based homomorphic cryptosystems as follows.

Definition 2. ***(Ring-based homomorphic encryption.)****An encryption scheme $\mathcal{S} = (Enc, Dec, \mathcal{K})$ is said to be ring homomorphic if for all $k \in \mathcal{K}$, it is possible to define groups $\mathcal{M}$ and $\mathcal{C}$ so that:*

- *$\forall m \in \mathcal{M} : c = Enc(m)$, it holds that $c \in \mathcal{C}$.*
- *$\forall m_1, m_2 \in \mathcal{M}, \forall c_1, c_2 \in \mathcal{C}$ with $m_1 = Dec(c_1)$ and $m_2 = Dec(c_2)$, it holds that:*

$$Dec(c_1 \times c_2) = Dec(Enc(m_1, k) \times Enc(m_2, k)) = m_1 \times m_2$$
$$Dec(c_1 + c_2) = Dec(Enc(m_1, k) + Enc(m_2, k)) = m_1 + m_2$$

where $\times$ and $+$ are the respective ring operations in $\mathcal{C}$ and $\mathcal{M}$.

Based on the rings, Gentry [Gentry 2009] introduces the first Fully Homomorphic Encryption (FHE) allowing to perform any number of additions and multiplications. Later on, there have been a lot of FHE schemes proposed [van Dijk et al. 2010, Stehlé and Steinfeld 2010, Brakerski and Vaikuntanathan 2011b, Brakerski and Vaikuntanathan 2011a, Vaikuntanathan 2011, Brakerski et al. 2012, Gentry et al. 2013] which improved the performance of the original FHE scheme scrupulously. However, up to now, FHE continues to be extremely slow for performing arbitrary functions or for implementing

the complex systems used today. This was illustrated through an evaluation of the AES circuit reporting that 40 minutes are needed to perform a single AES block on a machine with very large memory [Gentry et al. 2012]. In fact, two main factors make FHE inefficient: the cryptographic overhead and the used security definition. The cryptographic overhead represents the needed time to perform operations for each gate of the circuit implementing the program to evaluate. The security guarantees provided by FHE are to much strong in a way that makes some needed optimizations unenforceable.

2.3 Outsourced Data Confidentiality by Dissociation

Traditional solutions used to ensure the confidentiality of outsourced data are based on encryption. Unfortunately, the use of encryption makes performing search operations as well as other functionalities (e.g., computation) over the outsourced data costly. Nonetheless, if we look carefully to analyze the sensibility of the outsourced data, we realize that in most cases, mainly when dealing with outsourced relational databases, the associations between the information to be outsourced are more sensitive than the information themselves. As a consequence, many solutions based on data dissociation have been proposed. In the first part of this section, we will present and discuss proposed approaches based on data fragmentation to ensure the confidentiality of outsourced data. In the second part of this section, we will focus on approaches combining encryption and fragmentation to ensure outsourced data confidentiality.

2.3.1 Confidentiality by Fragmentation

Traditionally, data fragmentation techniques are aimed to enhance the data manipulation process, reducing the time needed to data processing by distributing data processing, optimizing data storage, etc [Randell 1969]. Nonetheless, data fragmentation based techniques are not designed with confidentiality preserving in mind.

Hudic et al. [Hudic et al. 2013] proposed an approach based on data fragmentation to protect outsourced relational databases confidentiality. The proposed solution is based on the use of a distribution model composed of two domains: a trusted local domain from where the data originates and a honest but curious public domain to where the data are distributed. The proposed fragmentation model relies on the classification of the relational tables composing the relational database to be outsourced

into different levels of confidentiality which are depending on the information that the respective relational tables store. Three different confidentiality levels are used: High Confidentiality tables, Medium Confidentiality Tables and, Low Confidentiality Tables. High Confidentiality tables store highly sensitive data, such as credit card numbers, personal identification numbers, which need to be protected appropriately. In order to minimize the use of encryption, High Confidentiality tables have to be stored in the trusted local domain without encryption. Medium Confidentiality Tables and Low Confidentiality Tables will be outsourced to different Cloud storage service providers.

Sai Krishna et al. [Krishna et al. 2012] propose a solution based on hybrid fragmentation, i.e., a combination of horizontal and vertical fragmentation to minimize the amount of data to be stored at the owner site. They have employed graph-coloring algorithms to determine which parts of a relational database can be outsourced and which parts need to be kept at the owner.

Clearly, [Hudic et al. 2013] and [Krishna et al. 2012] are not good approaches since the data owner must always manage and protect the relational tables containing highly sensitive information.

2.3.2 Confidentiality by Combining Fragmentation and Encryption

The use of data fragmentation to enforce outsourced data confidentiality has been first proposed, in conjunction with encryption, by Aggarwal et al. [Aggarwal et al. 2005]. The main idea consists of allowing the data owner to fragment its data across two or many Cloud storage service providers that cannot communicate with each other. The data fragmentation is performed in such a way as to be sure that the disclosure of the contents of any one fragment does not lead to violate the confidentiality contraints. Then, to perform queries over the fragmented and distributed database, a client sends appropriate sub-queries to each Cloud storage service provider that stores a fragment of the original database, and then piecing together sub-queries results at the client side. The main drawback of this approach is that it is extremely hard in the real word to ensure that the Cloud storage service providers to which the data is distributed cannot communicate with each other.

Following the same idea proposed in [Aggarwal et al. 2005], Ciriani et al. [Ciriani et al. 2007, Ciriani et al. 2009] define an approach considering that associations between a relational table attributes as well as the values assumed by some of them can be sensitive. They define two kinds of constraints to allow the data owner

to specify his/her confidentiality requirements: Singleton constraints and Association constraints. A singleton constraint is used to specify that the values assumed by an attribute are sensitive. An association constraint is used to specify that the associations between two or more attributes are sensitive and must be protected. The fragmentation mechanism used is as following. Consider a set of confidentiality constraints and a set of attributes $\mathcal{A}$ that should be fragmented (all attributes belonging to the database to be outsourced, except those concerned by singleton constraints). The result of the fragmentation is represented by a set of fragments $\mathcal{F} = \{F_1, \cdots, F_n\}$ where each fragment should verify three properties: (1) Ensure that only the attributes in $\mathcal{A}$ are concerned by the fragmentation, (2) ensure that each attribute in $\mathcal{A}$ appears at least one time in plaintext in a fragment, and (3) guarantee the unlinkability between the fragments in $\mathcal{F}$ (no plaintext attribute in common between the fragments in $\mathcal{F}$). The authors show that the satisfaction of confidentiality constraints while respecting the three previously mentioned properties makes the fragmentation problem so far from being trivial. Moreover, they prove that satisfy the confidentiality constraints while minimizing the number of fragments (in order to avoid the unnecessary fragmentation of attributes) makes the fragmentation problem NP-complete [di Vimercati et al. 2010]. To overcome this problem, Ciriani et al. in [Ciriani et al. 2012] proposed a new modeling of the fragmentation problem that exploits the compact representation of confidentiality constraints as Boolean formulas, which makes finding a solution to the fragmentation problem relies on the efficiency of representation and manipulation of those Boolean formulas. To meet this requirements, the authors have used Ordered Binary Decision Diagram (OBDDs) allowing an efficient manipulation of Boolean formulas.

Unfortunately, the approaches proposed in [Ciriani et al. 2007, Ciriani et al. 2009] suffer from a major limitation since they suppose that the data to be outsourced are stored in only one relational table. Clearly, this hypothesis is seldom in the real production environments.

2.4 Conclusion

Outsourced data confidentiality is becoming an emerging paradigm that introduces many research challenges. In this chapter, we presented and discussed existing solutions related to ensuring outsourced data confidentiality. Two simple lessons can be taken from the above discussions: (1) There are several ways to protect the confidentiality of outsourced data providing different tradeoffs in the space of security, functionality, and efficiency. (2) When solutions providing strong security guarantees (e.g., homomorphic schemes) cannot provide a practical solution, we should take advantage of controlled

leakage solutions. That is, choosing the information to be revealed to the Cloud storage service provider in a way that enables search optimizations, while ensuring a meaningful confidentiality guarantee.

In the next chapter, we propose an approach allowing the protection of confidentiality of sensitive information in outsourced multi-relational databases by improving existing approaches [Ciriani et al. 2007, Ciriani et al. 2009] based on combining data fragmentation together with encryption.

CHAPTER 3

Preserving Multi-relational Outsourced Databases Confidentiality using Fragmentation and Encryption

3.1 Introduction

Database as a service models give rise to two significant challenges. The first challenge is how service providers protect outsourced databases from not authorized users. A straightforward solution to protect outsourced databases consists in encrypting data before their outsourcing. Unfortunately, it has been previously mentioned in Chapter 2 that querying data becomes in this case expensive (heavy computational overheads) and can be impossible for several kind of queries. The second challenge is more complex as it concerns the protection of outsourced databases from the storage service providers, as in this case, they are not considered to be fully trusted. Therefore, our main focus in this chapter is to define an approach that preserves outsourced data confidentiality while providing a secure and efficient querying technique.

3.1.1 Motivating Scenario

In our working scenario, we strive to protect the confidentiality of an outsourced relational hospital database $\mathcal{D}$ composed of two relations (primary keys are underlined

and foreign keys indicated by *) :

Patient(Id_P, Name,ZIP, Illness, Id_Doctor *)
Doctor(Id_D, Name, Specialty)

The relationship between the tables *Patient* and *Doctor* is defined between the foreign key of the table *Patient* (*Id_Doctor*) and the primary key of the table *Doctor* (*Id_D*). We assume that the database will be outsourced to a third party. Therefore, sensitive information stored in $\mathcal{D}$ should be protected. One classic solution is to encrypt all information before outsourcing the database, a costly operation. However, if we look carefully, the list of patients and their attributes ($Id\_P, Name, Zip$) can be considered as insensitive, and also that the list of illnesses could be made public. Nevertheless, data sensitivity arises from the relationship between these two lists (list of patients and list of illnesses). Therefore if we can find a way (e.g. vertical fragmentation [Navathe et al. 1984]) to break relationships between patients and their respective illnesses, there is no need to encrypt all records of the *Patient* relation. On the other hand, the list of doctors and the list of patients are not sensitive. However, the relationship between a patient and his doctor should be protected. The good way to protect the relationship between the two relations *Patient* and *Doctor* consists in encrypting the foreign key *Id_Doctor* or the primary key *Id_D*. The encryption of the foreign key appears to be more beneficial as a foreign key references only one relation (only the relationship between the two relations is protected) while a primary key can be referenced by many relations. Therefore, if we encrypt the primary key, we will protect all relationships between the relation containing the primary key and other relations referencing the encrypted primary key. Thus, when the security requirement specifies that only the a relationship between data is sensitive, our apporach is more appropriate than the one based on full encryption.

3.1.2 Contributions

In this chapter, we propose an approach to protect the confidentiality of sensitive outsourced databases by combining the best features of fragmentation and encryption. Furthermore, we present an approach which is able to deal efficiently with multi-relation normalized databases with which we strive to overcome the previously mentioned limitations of [Ciriani et al. 2007, Ciriani et al. 2009]. The problems encountered in one-relation[1] databases take on additional complexity when working with multi-relation normalized databases in a distributed environment, as it gives rise to new problems

[1]Databases composed from a single relation schema.

such as protecting the relationships between relational schemas (relationships between tuples in distinct tables) and defining a secure and efficient technique allowing authorized users to query these sensitive relationships. We will present our approach which uses a practical Private Information Retrieval (PIR) technique [Chor et al. 1998] allowing to protect data unlinkability of different fragments of the original database by protecting user query privacy. Unlinkability of two items of interest (e.g., records stored into different fragments) means that within the system, from an adversary point of view, these items of interest are no more and no less related. In our approach, a relation containing sensitive information will be fragmented vertically into two or more fragments. Unlinkability of fragments means that despite the fact that an adversary has knowledge about the fragments of a relation, he/she remains unable to link records from different fragments. Furthermore, we evaluate our proposed protocol by presenting some experiments using our developed prototype. Afterwards, we use hash table data structures to store the information of each fragments instead of using B+ trees which allows us to improve the effectiveness of the proposed PIR keyword-based technique.

3.1.3 Chapter Outline

We proceed by describing in Section 3.2 our model architecture, the threat model, security model and assumptions. After that, we describe in Section 3.3 our approach to enforce confidentiality of outsourced data. Section 3.4 presents the query optimization and execution model. In Section 3.5, we present a PIR-based technique to achieve query privacy and enforce data confidentiality under a collaborative Cloud storage service providers model. In Section 3.6, we present the prototype developement and experimentations we conducted. Finally, Section 3.7 concludes this chapter.

3.2 Technical Preliminaries

3.2.1 Architecture

We consider our architecture of storage and query over distributed fragments illustrated in Figure 3.1. It is composed of three main components:

- **Users**: They are actually database clients who have permission to query outsourced data. All operations which will be used in our approach (e.g., fragmentation and encryption) in order to protect sensitive data confidentiality are trans-

parent to the *Users*. That is, they believe interacting the original database and forming their queries against it.

- **Client**: It is a trusted party which transform *Users* queries by splitting them to create an optimized distributed Query Execution Plan QEP; QEP is a set of sub-queries and other operations (e.g., decryption, join...). Based on the *Metadata* containing information (e.g., relations, clear attributes, encrypted attributes and selectivity [2] of attributes) about data distribution in different fragments, the *Query Transformation module* construct a QEP which will be executed.
- **Server**: It represents different Cloud Storage Providers in which data fragments are distributed.

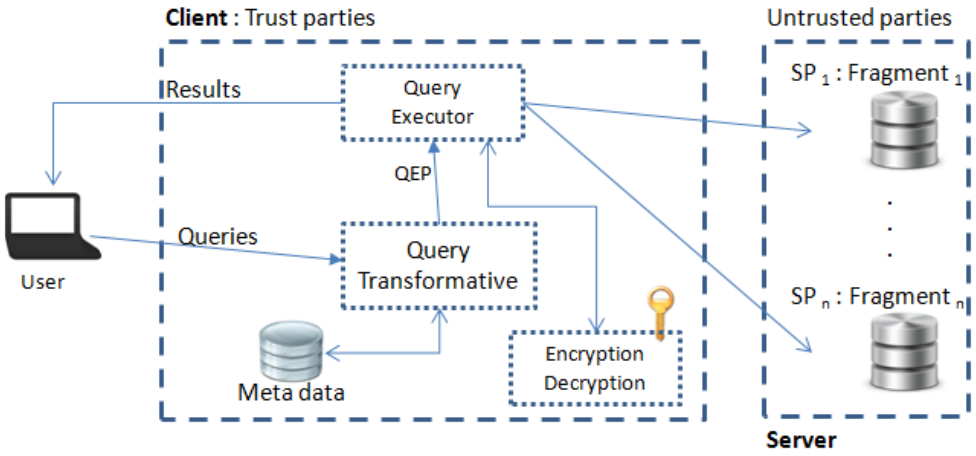


Figure 3.1 – Architecture of the Proposed Model

3.2.2 Trust and Attack Model

Cloud servers are considered to be the best options for small companies with limited IT budget allowing to reduce the cost of maintaining computing infrastructure and data-rich applications. However, most of related works (e.g.,[Hacigümüs et al. 2002, Biskup et al. 2011]) on the confidentiality of outsourced data considered that Cloud service providers are "honest-but-curious". The *semi-honest* model is the right fit for our approach, as in this model, the Cloud servers act in an "honest" manner by correctly responding user queries and following the designated protocol specification. In this contribution, we consider that Cloud services providers have two levels of curiosity: (1) In the first part of this chapter, we will assume that service providers are "curious"

[2] Attribute selectivity is is an estimated number that determines the effectiveness of queries that performs a search over this attribute.

in a way that they will try to infer and analyze outsourced data, and will also actively monitor all received user queries and try to derive as much information as possible from these queries. (2) In the second part of this chapter, we will further assume that service providers can collude and cooperate together to link outsourced data. The client part of this architecture is assumed to be trustworthy and all interactions between the user and the client are secured using existing protocols e.g., SSL.

3.3 Confidentiality Using Fragmentation and Encryption

Our approach extends in several ways the vertical fragmentation-based approach described in [Ciriani et al. 2007, Ciriani et al. 2009]. This approach considers that all data is stored in a single relation, while in our approach data can be stored in several relations, which is the rule for any typical environments. In our approach, we consider that databases to be externalized are normalized so that two relations can be only associated together through a primary key/foreign key relationship. For this purpose, we introduce a new type of confidentiality constraint for fragmentation, the *inter-table fragmentation constraint*. The aim of this new fragmentation constraint is to protect the relationship between relations.

This section first presents the different kinds of confidentiality constraints used to achieve our goals of protecting confidentiality by encryption and fragmentation, and second formalizes the concept of fragmentation in our approach which extends ideas presented in [Ciriani et al. 2007, Ciriani et al. 2009].

Definition 3. ***(Confidentiality Constraint).*** *Consider that data to be secured are represented with a relational database $\mathcal{D}$, which is composed of a list of relational schemas $R = (R_1, \ldots, R_n)$, with each of these relational schemas R_i containing a list of attributes $A_{R_i} = (a_{1,i}, a_{2,i}, \ldots)$. A confidentiality constraint over $\mathcal{D}$ can be one of the following:*

- ***Singleton Constraint** (SC).* *It is represented as a singleton set $SC_{R_i} = \{a_{j,i}\}$ over the relation R_i. This kind of confidentiality constraint means that the attribute $a_{j,i}$ of the relational schema R_i is sensitive and must be protected, typically by applying encryption.*
- ***Association Constraint** (AC).* *This kind of confidentiality constraint is represented as a subset of attributes $AC_{R_i} = \{a_{1,i}, \ldots, a_{j,i}\}$ over the relational schema*

R_i. *Semantically, it means that the relationship between attributes of the subset* AC_{R_i} *is sensitive and must be protected.*

- ***Inter-table Constraint*** *(*IC*).* *It is represented as a couple of relational schemas* $IC = \{R_i, R_j\}$ *of the relational database* $\mathcal{D}$. *Relations* R_i *and* R_j *must be associated through a primary key/foreign key relationship. The use of this kind of confidentiality constraint ensures protection of the primary key/foreign key relationship between the two relational schemas concerned with the inter-table constraint* IC.

Note that protecting the relationship between two tables relies on protecting the primary key/foreign key relationship and storing involved relations on distinct servers of distinct providers. The association constraint can also be addressed through encryption (encrypt at least one of attributes involved in the constraint), but clearly this will increase the number of encrypted attributes and make database interrogation more complicated. A more adapted way to resolve this kind of confidentiality constraint was proposed in [Ciriani et al. 2007], which is based on splitting involved attributes in a manner that their relationships cannot be reconstructed.

In the case of an inter-table confidentiality constraint, protecting the foreign key using encryption is the simplest way to secure the relationship between the two relational schemas. However encrypting only the foreign key is not enough to keep the relationship between relational schemas secure, as service provider may be able to link records in two relational schemas by observing and analyzing user queries over these relational schemas. To overcome this problem, the two relational schemas involved in that case should be split into different fragments, and each of these fragments should be distributed to a different Cloud storage provider. An interesting approach for modeling constraints and resolving the data fragmentation problem was proposed in [Ciriani et al. 2012], that efficiently computes data fragments satisfying the confidentiality constraints. It is based on Boolean formulas and Ordered Binary Decision Diagrams (OBDD) and uses only attribute-based confidentiality constraint (Singleton Constraints and Association Constraints). However, it cannot deal as-is with Inter-table Constraints. In order to use this approach, we define a way to reformulate Inter-table Constraint as a set of Singleton Constraints and Association Constraints. We explain this transformation in the definitions and theorems below.

Definition 4. ***(Inter-table Constraint transformation).*** *Consider a relational database with two relations* $R_1(\underline{a_1}, \ldots, a_n)$ *and* $R_2(\underline{b_1}, \ldots, b_m*)$. *Let us assume that* R_1 *and* R_2 *are related through a foreign key/primary key relationship in which the foreign key* b_m *of the relation* R_2 *references the primary key* a_1 *of relation* R_1. *We assume that* R_1 *and* R_2 *contain respectively* p *and* q *records, with* $p > 1$ *and* $q > 1$. *An Inter-table*

Constraint $c = \{R_1, R_2\}$ over relations R_1 and R_2 states that the relationship between these two relations must be protected by encrypting the foreign key b_m and by storing R_1 and R_2 in two different fragments. Therefore, the constraint c can be written as follows:

1. *A singleton constraint $SC = \{b_m\}$ to state that the value of b_m should be protected.*
2. *A list of $(m \times n)$ association constraints $AC = \{(a_i, b_j) | i \in [1, n],\ j \in [1, m]\}$.*

We propose the notion of a correct transformation of Inter-table Constraints. A transformation of an Inter-table Constraint c to a set of confidentiality constraints C is correct if enforcement of C implies protection of the unlinkability between records of the two relations involved in c. The following Theorem formalizes this concept.

Theorem 1. ***(Transformation correctness).*** *Given a relational database $\mathcal{D}$ made up of two relational schemas $R_1(\underline{a_1}, \ldots, a_n)$ and $R_2(\underline{b_1}, \ldots, b_m*)$ related through relationship between the foreign key b_m of R_2 and the primary key a_1 of R_1. Let $c = \{R_1, R_2\}$ be an Inter-table Constraint, the set of constraints C be the result of the transformation of c, and $\mathcal{F} = \{F_1, \ldots F_q\}$ be a fragmentation of $\mathcal{D}$ that satisfies C. The Inter-table Constraint c is correctly transformed into a set of constraints C if all the following conditions hold :*

1. *b_m does not appear in clear in any fragment of $\mathcal{F}$.*
2. *$\forall\ AC_{i,j} = \{a_i, b_j\} \in C, i \in [1, n], j \in [1, m],\ if\ \ a_i \in F_k\ \ and\ \ b_j \in F_l\ \ then\ k \neq l$.*

Proof. According to Item 2 of Definition 4, the Inter-table Constraint will be replaced by all possible associations constraint composed from an attribute of relation R_1 and another from relation R_2. Due to the fact that an association constraint between two attributes means that the relationship between these attributes will be protected using fragmentation (each attribute will be stored in different fragments), Item 2 guarantees that relations R_1 and R_2 will be stored in different fragments which hold condition (2).

Item 1 of Definition 4 creates a singleton constraint over the foreign key b_m of the relation R_2. Thus b_m will be considered as a sensitive attribute and will be protected using encryption, which means that the foreign key b_m will not appear in clear in any fragment. As a result, if an adversary succeeds in having access to the fragments in which R_1 and R_2 have been stored, she is unable to link data stored in these relations. □

The main advantage of the Inter-table Constraint is that it allows treatment of multi-table relational databases. In addition, it gives a simple way to formulate confidentiality constraints between relations. As we have seen in Item 1 of Definition 4, the attribute b_m (foreign key of the relation R_2) should be encrypted. However, to be able to query data and construct relationship between relations, the chosen encryption algorithm must be deterministic [Bellare et al. 2008] in order to preserve uniqueness and allow the construction of relationship between relations (e.g. through JOIN queries). As is known, in normalized multi-relation databases, three types of relationship between relations exist: (i) one-to-one, (ii) one-to-many and (iii) many-to-many relationships. Inter-table Constraints over relations associated using (i) or (ii) can be simply transformed as shown in Definition 4, while others associated using (iii) need a pre-transformation step before applying the transformation of Definition 4, as they are normally linked through a third relation known as a linking table. The pre-transformation steps is described in the example below.

Example 1. *Consider that we have a hospital relational database $\mathcal{D}$ with relations :*

Patient$(\underline{Id\_patient}, Name, ZIP)$
Doctor$(\underline{Id\_doctor}, Name, Specialty)$
Examination$(\underline{Id\_examination}, date, medical\_report, Id\_doctor*, Id\_patient*)$

Assume that database owner claims that relationships between a patient and his/their doctor(s) are sensitive and must be secured. Therefore an Inter-table Constraint over relation Patient and Doctor ($IC = \{Patient, Doctor\}$) must be defined. In this case applying directly transformation as shown in Definition 4 is not possible since relations Patient and Doctor are connected through Examination. So, the pre-transformation step consists in writing the Inter-table Constraint IC using the linking relation Examination. Thus, IC will be replaced by $IC_1 = \{Patient, Examination\}$ and $IC_2 = \{Doctor, Examination\}$. Next, both IC_1 and IC_2 will be transformed into a set of Singleton Constraints and Association Constraints according to Definition 4.

Definition 5. ***Fragmentation.*** *Let us consider a relational database $\mathcal{D}$ with relations $R_1, \ldots, R_n$ and A the list of all attributes contained in these relations. Given A_f the list of attributes to be fragmented, the result of the fragmentation is a list of fragments $F = \{F_1, \ldots, F_m\}$ where each of these fragments satisfies:*

1. $\forall F_i \in F, i \in [1..m],\ F_i \subseteq A_f$.
2. $\forall a \in A_f, \exists F_i \in F : a \in F_i$.
3. $\forall F_i, F_j \in F, i \neq j : F_i \cap F_j = \emptyset$.

Note that the list of attributes to be fragmented A_f contains all attributes in A, except those concerned with Singleton Constraints (attributes to be encrypted). Condition 1 guarantees that only attributes in A_f are concerned by the fragmentation, condition 2 ensures that any attribute in A_f appears in clear at least in one fragment and condition 3 guarantees unlinkability between different fragments.

Logically, to be able to get information about the original database, we should be able to reconstruct original database from fragments. So after defining the fragmentation process, we shall define a mechanism to combine fragmentation and encryption. More precisely, we need a mechanism to integrate attributes involved in the Singleton Constraints (attributes to be encrypted) in the suitable fragment. These encrypted attributes allow only authorized users (users who know the encryption key) to construct the sensitive relationships. Based on the definition of $Physical fragment$ proposed in [Ciriani et al. 2007], we define our mechanism called $Secure fragment$ to combine fragmentation and encryption.

Definition 6. ***(Secure Fragment).*** *Let $\mathcal{D}$ be a relational database with a list of relations $R = \{R_1(a_{1,1}, \ldots, a_{j,1}), \ldots, R_n(a_{1,n}, \ldots, a_{k,n})\}$, $F = \{F_1, \ldots, F_m\}$ a fragmentation of D and A_f be the list of fragmented attributes. Each fragment $F_i \in F$ is a new relation whose attributes are a subset $A_i \subseteq A_f$. Each A_i is composed of a subset of attributes of one or more relations $R_j \in R$. We denote by R_{F_i} the list of relations in R such that a subset of their attributes belongs to the fragment $F_i \in F$. The secure fragment of F_i is represented by a set of relations schema $R^e_{F_i}$ in which each relation is represented as $R^e_j(\underline{salt}, enc, a_1, \ldots, a_k)$ where $\{a_1, \ldots, a_k\} \subset A_i \cap R_j$ and enc is the encryption of all attributes of R_j that do not belong to $\{a_1, \ldots, a_k\}$ (all attributes of R_j involved in a singleton constraint except those concerned by a singleton constraint over the foreign key), combined before encryption in a binary XOR with the salt. All foreign key attributes which are involved in singleton constraints are encrypted using a deterministic encryption algorithm (e.g., AES) to ensure their distinguishability.*

Algorithm 1 shows the construction of secure fragments.The main reason for reporting all original attributes (except foreign keys involved in the Singleton constraints) in an encrypted form for each relation in a fragment, is to guarantee that a query Q over the original relation R_j can be executed by querying a single fragment (which contains R^e_j) while preserving confidentiality of sensitive relationships, so we do not need to reconstruct the original relation R_j to perform the query Q. Furthermore, encrypting foreign keys ensure the protection of sensitive relationships between relations involved into Inter-table Constraints. However, using deterministic encryption algorithm has two issues. First, a major advantage is to enforce indistinguishability of records which allows only authorized users who know the encryption key to execute queries associat-

input : $\mathcal{D} = \{R_1, R_2, \cdots, R_n\}$ /* Normalized relational database */
$\mathcal{C} = \{C_1, C_2, \cdots, C_m\}$ /* Confidentiality constraints */
output: $\mathcal{F}^s = \{F_1^s, F_2^s, \cdots, F_p^s\}$ /*The set of secure fragments*/

$\mathcal{C}_f = \{C_i \in \mathcal{C} : |C_i| > 1\}$ /* The list of association constraints */
$\mathcal{A}_{fkey} = \{a \in C_i, C_i \in \mathcal{C} : |C_i| = 1 \; and \; \textbf{isForeignKey}(a) = True\}$
/* $\mathcal{A}_{fkey}$: The set of foreign keys to be encrypted*/

/* Fragmentation */
$\mathcal{F} := \textbf{Fragment}(\mathcal{D}, \mathcal{C}_f)$
foreach $F_i = \{a_{i_1}, a_{i_2}, \cdots, a_{i_n}\}$ *in* $\mathcal{F}$ **do**
    $\mathcal{R}_f = \textbf{classifyAttributes}(F_i)$ /* Classify the attributes according to their original relation.*/
    foreach R_{f_i} *in* $\mathcal{R}_f$ **do**
        foreach r *in* R_{f_i} **do**
            /* r : record */
            $r^s[salt] := \textbf{GenerateSalt}(R_{f_i}, r)$
            $r^s[enc] := \mathcal{E}_k(t[a_{j_1}, \cdots, a_{j_q}] \oplus r^s[salt])$
            /* $a_{j_1}, \cdots, a_{j_q} = R_i - R_{f_i}$ */
            foreach a *in* R_{f_i} **do**
                /* a : attribute */
                $r^s[a] := r[a]$
            endfch
            foreach a *in* A_{fkey} **do**
                if $a \in R_i$ **then**
                    /* a : the foreign key of the relation R_i */
                    $r^s[a] := \mathcal{E}_k(r[a])$
                end
            endfch
            InsertRecord(r^s, R^s)
        endfch
        AddRelationToFragment(R^s, F^s)
    endfch
endfch

Algorithm 1: Secure fragmentation

ing these relations. Second, a minor drawback is that it allows an adversary to infer information about repeatedly occurring values of the encrypted foreign keys, but this

information does not allow the adversary to break the unlinkability between relations. The attribute *salt* which is used as a primary key of different relations in the secure fragments protects encrypted data against frequential attacks. In addition, there is no need to secure the *salt* attribute because knowledge of the value of this attribute will not give any advantage when attacking encrypted data. In fact, the algorithm we used ($AES(k, m \oplus salt)$) can be seen as an AES-CBC with an $IV = salt$, or AES-CBC are proved to be semantically secure [Bellare et al. 1997] even when the used IV is known.

Example 2. *Assume that we have a relational database $\mathcal{D}$ of a medical insurance company that contains two relations Patient and Doctor represented respectively in Table 5.1 and Table 5.2. The insurance company has defined a set of confidentiality constraints* $CC = \{C_1 = \{SSN\}, C_2 = \{Name\_pat, Illness\}, C_3 = \{Patient, Doctor\}\}$.

Table 3.1 – Patient relation

SSN	Name_pat	Dob	Illness	Id_doc
865746129	A. Barrett	20-08-1976	Illness_1	doc_3
591674603	C. Beat	18-01-1981	Illness_2	doc_3
880951264	N. Baines	14-09-1986	Illness_1	doc_2
357951648	S. Brandt	18-01-1981	Illness_3	doc_1

Table 3.2 – Doctor relation

Id_doctor	Name_doc
doc_1	C. Amalia
doc_2	D. Annli
doc_3	P. Amadeus

As shown before, the first step in the fragmentation process consists in transforming Inter-table Constraint (C_3). Relations Patient and Doctor are linked through the foreign key Id_doc in the relation Patient, therefore C_3 will be replaced by $C_4 = \{Id\_doc\}$ and all possible Association constraints composed of an attribute of the relation Doctor and an attribute of the relation Patient (guarantee that the relation Patient will not be in the same fragment as the relation Doctor). In our example, attributes SSN and Id_doc of the relation Patient are involved in singleton constraints C_1 and C_4 respectively. So they will not be concerned by the fragmentation. As a result C_3 will be replaced by :

- $C_4 = \{Id\_doc\}$
- $C_5 = \{Name\_pat, Id\_doctor\}$
- $C_6 = \{Name\_pat, Name\_doc\}$
- $C_7 = \{Dob, Id\_doctor\}$
- $C_8 = \{Dob, Name\_doc\}$
- $C_9 = \{Illness, Id\_doctor\}$
- $C_{10} = \{Illness, Name\_doc\}$

A possible fragmentation of $\mathcal{D}$ that satisfies all confidentiality constraints is the set of fragments $\{F_1, F_2, F_3\}$ with: $F_1 = \{Patient(Name\_pat, Dob)\}$, $F_2 = \{Patient(Illness)\}$ and $F_3 = \{Doctor(Id\_doctor, Name\_doc)\}$. Next step is the Secure fragmentation transformation (Definition 6). We assume that encryption of the protected attributes uses the deterministic encryption algorithm E with the encryption key K. The result of applying the $SecureFragmentation$ over different fragments is represented as follows.

- $F_1 : Patient(\underline{salt}, enc, Name\_pat, Dob, E_k(Id\_doc))$ with:
$$enc = E_K(\langle SSN, Illness\rangle \oplus salt)$$
- $F_2 : Patient(\underline{salt}, enc, Illness, E_k(Id\_doc))$ with:
$$enc = E_K(\langle SSN, Name\_pat, Dob\rangle \oplus salt)$$
- $F_3 : Doctor(Id\_doctor, Name\_doc)$

Figure 3.2 – Secure Fragmentation Results

Note that F_3 has not been changed because there is no singleton constraints over the Doctor attributes. Lastly data fragments F_1, F_2 and F_3 are distributed to different Cloud storage providers.

3.4 Query Transformation and Optimization

In our querying model, query transformation is performed by the *Query Transformation* (QT) module on the client side. When receiving a user query, the query is analyzed syntactically and semantically so that incorrect queries are rejected as earlier as possible. Next, based on the $Metadata$ stored on the client side, the QT will attempt to find a fragment on which the user query can be executed, i.e. a fragment in which QT can find all attributes and relations involved in the user query. If such a fragment does not exist, QT will decompose the user query into queries expressed in relational algebra, find out which fragments are involved in the query, and finally transform the user query into a set of fragments queries. Using this set of fragment queries and other operations such as encryption, decryption, join and aggregation, the QT creates

```
   input : Q      /* User Query */
           M      /* Metadata */
   output: QEP    /*Query execution plan*/
1
2  (tables, attributes, conditions) = decomposeQuery(Q)
3  syntacticChecking(Q)  /* Verify that keywords, object names, operators are placed correctly in the query*/
4  /* Semantic Checking */
5  if tables ⊈ M or attributes ⊈ M then
6  |  rejectQeury(Q)
7  end
8  foreach (attribute, operator, value) in conditions do
9  |  if !isTheSameType(attribute, value) or !match(operator, value) then
10 |  |  rejectQeury(Q)
11 |  end
12 endfch
13 /* Checking if there is a fragment on which the query can be directly executed */
14 F = getFragmentSchema(M)
15 foreach frag in F do
16 |  T_frag = getTables(frag)
17 |  A_frag = getAttributes(frag)
18 |  A_c = getAttributesFromConditions(Conditions)
19 |  if tables ⊈ T_frag or attributes ⊈ A_frag or A_c ⊈ A_frag then
20 |  |  continue
21 |  end
22 |  /* Checking if all conditions attributes are not encrypted in the fragment frag*/
23 |  if areEncrpted(A_c, frag) then
24 |  |  continue
25 |  end
26 |  /*The query Q can be executed on the fragment frag */
27 |  addOperation(QEP, (Q,frag))
28 |  foreach attr in A_frag do
29 |  |  if isEncrpted(attr, frag) then
30 |  |  |  addOperation(QEP, (Decryption,attr))
31 |  |  end
32 |  endfch
33 |  return QEP
34 endfch
35
36 /* Multi Fragment Query*/
37 /* Get the fragments in which conditions attributes are not encrypted */
38 A_c = getAttributesFromConditions(Conditions)
39 A_c^s = sortAttributeBySelectivity(M, A_c) /* Sort attributes according to their selectivity*/
40 foreach attr in A_c^s do
41 |  F_attr = containsInClear(M, attr) /* Set of fragments on which attr appears in clear text*/
42 |  frag = getBestFragment(F_attr) /* Get the best fragment which contains the less number of encrypted
   |  attributes */
43 |  A = listOfRetrievedAttributes(frag, attributes) /* The list of attributes that can be retrieved by
   |  querying the fragment frag */
44 |  SQ = formulateTheSubQuery(A, attr, Q)
45 |  addOperation(QEP, (Q, frag))
46 |  foreach a in A do
47 |  |  if isEncrpted(a, frag) then
48 |  |  |  addOperation(QEP, (Decryption, a))
49 |  |  end
50 |  endfch
51 endfch
52 /*Add the join operation that combines results returned from subqueries*/
53 addOperation(QEP, join)
```

Algorithm 2: Query validation and transformation process

a QEP and sends it to the *Query Executor*. Algorithm 2 shows the query validation, transformation and optimization process.

Example 3. ***(One-fragment query)***. *Assume that we have a relational database $\mathcal{D}$ that contains two relations Patient and Doctor represented respectively in Table 3.1 and Table 3.2, The fragmentation of $\mathcal{D}$ is the list of fragments represented in Figure 3.2. Consider the following user query:*

$Q1$:
```
SELECT Name_pat, SSN
FROM patient
WHERE Dob='1986-09-14'
And Illness = 'Illness1';
```

$Q1$ can be executed over either F_1 or F_2 fragments as all attributes required by $Q1$ can be found (in clear or encrypted form) in both fragments.

- *QEP_1 for $Q1$ over F_1 :*

$Q1_1$:
```
SELECT Name_pat, salt, enc
FROM patient
WHERE Dob='1986-09-14';
```
Dec : $Decrypt(Result(Q1_1), Key) = R^d(Q1_1)$

$Q1_2$:
```
SELECT Name_pat, SSN
FROM R^d(Q1_1)
WHERE Illness = 'Illness1';
```

- *QEP_2 for $Q1$ over F_2 :*

$Q1_1$:
```
SELECT salt, enc
FROM patient
WHERE Illness = 'Illness1';
```
Dec : $Decrypt(Result(Q1_1), Key) = R^d(Q1_1)$

$Q1_2$:
```
SELECT Name_pat, SSN
FROM R^d(Q1_1)
WHERE Dob='1986-09-14';
```

As we can see through the previous example, a query can have more that one QEP. Logically, each QEP may have a different execution cost. Thus, the *QT* should have the capability to pick out the best QEP in terms of execution cost. This capability is explained later in the Query Optimization section.

For multi-fragment query[3], QT will use local join operations as it should combine results of execution of subqueries over fragments. There are two different ways to perform local join operation : (1) Execute all sub-queries in a parallel manner, then join the result on the client side. (2) Execute sub-queries in a sequential manner to have the ability to perform *semi-joins* using the result of previous sub-queries. While (1) can be cheaper than (2) in terms of sub-query execution, it is much more costly in the join operation because in (1), sub-queries results might contain a lot of records that might not be part of the final results.

Example 4. ***(Multi-fragment query).*** *Assume that we will use the same database $\mathcal{D}$ and fragments used in Example 3. Consider the query :*

```
Q2 :  SELECT Name_pat, Name_doc
      FROM patient, doctor
      WHERE Dob='1986-09-14'
```

A possible QEP for Q2 can be :

$Q2_1(F_1)$: `SELECT Name_pat,` $E_k(Id\_doc)$
`FROM patient`
`WHERE Dob = '1986-09-14';`

Dec : $Decrypt(E_k(Id\_doc), Key) = \delta$

$Q2_2(F_3)$: `SELECT` $Name\_doc$
`FROM doctor`
`WHERE` Id_doctor `IN` δ`;`

`Join :` `Result(`$Q2_1$`)` $\bowtie$ `Result(`$Q2_2$`)`

Since the relationship between the two relations *Patient* and *Doctor* is protected, these relations are stored in different fragments. Therefore, the query $Q2$ is decomposed into two sub-queries $Q2_1$ and $Q2_2$ executed respectively over fragments F_1 and F_3.

In addition to traditional query optimization methods such as selecting conditions as earlier as possible, the QT attempts to minimize the execution cost of the created QEP by applying the selection condition with the most selective attribute, i.e the selection condition which is satisfied by the smallest number of tuples. To give this ability to the QT, we assign a selectivity and an average attribute-value size (AVS) to each attribute contained in the original database to the $Metadata$ stored in the *Client*. The selectivity of an attribute is the ratio of the number of distinct values to the total number of rows.

$$Selectivity = \frac{DistinctValues}{TotalNumberRows} \tag{3.1}$$

[3]i.e. a query that cannot be executed over only one fragment.

In distributed databases, they may exist several strategies for each query due to the fact that data are stored in different sites. One way to choose the best strategy is based on calculating the expected cost which should include corresponding evaluation and communication cost. The formula of a point query execution costs can be estimated roughly as follows:

$$Query\ execution\ cost = C_E \times NbExRow + C_T \times NbEstRow \tag{3.2}$$

C_E represents the evaluation cost of a record, C_T is the transmission cost of a record, $NbExRow$ is the number of rows examined and $NbEstRow$ is for the estimated number of returned rows which is calculated as follows :

$$NbEstRow = \frac{1}{Selectivity} \tag{3.3}$$

Using the average attribute-value size (AVS) of encrypted attribute *enc*, we can estimate the execution costs of the decryption operation as follows :

$$Decryption\ cost = C_D \times AVS \times Number\ of\ rows \tag{3.4}$$

C_D represents an estimation of the per-byte decryption costs of the used encryption schemes.

Example 5. *Assume that we use the same database $\mathcal{D}$ and fragments of Example 3. Let us suppose that the relation patient contains 10^5 tuples and the selectivity estimation of the attribute Dob is 0.14 and for Illness it is 8×10^{-4}. We suppose also that $AVS_1 = 252$ and $AVS_2 = 152$ are respectively the average attribute-value sizes of encrypted attribute enc of the table patient stored in the fragments F_1 and F_2. Consider the query $Q1$ used in the Example 5.1. As shown before, there are two possible QEP for this query. Using (2), (3) and (4) the QT will compute the approximative execution cost for each QEP as shown below :*

$$QEP_1\ execution\ cost = C_E \times 10^5 + C_T \times 7 + C_D \times 252 \times 7 + C_E \times 7$$
$$QEP_2\ execution\ cost = C_E \times 10^5 + C_T \times 1250 + C_D \times 152 \times 1250 + C_E \times 1250$$

After computing the approximative execution cost of each QEP, the QT will select the best one in terms of execution cost. In our example, QEP_1 has the lowest execution cost.

3.5 Preserving Data Unlinkability

Ensuring data confidentiality is achieved by preserving unlinkability between different data fragments and by encrypting all sensitive information that cannot be protected

using only fragmentation. However, we have seen in the previous section that evaluation of some queries may use *semi join* in order to join data from different fragments. This will not be a concern in the case of non-colluding Cloud providers, but it becomes a serious security and privacy problem when Cloud Service Providers (CSP) can collude. In this section, we present our solution to overcome this privacy concern when we assume that CSP can collude to link data stored in different fragments.

Example 6. *Consider the database, fragments, queries and QEP used in the Example 4. The QEP is executed in a sequential manner by the QueryExecutor. The next table shows the execution of the QEP.*

Operation	*Result*
$Q2_1$ *execution over* F_1 :	$\langle N.Baines, E_k(doc\_2)\rangle$
Decryption :	$\delta = \{Decryption(E_k(doc\_2))\}$
$Q2_2$ *execution over* F_3 :	$\langle D.Annli\rangle$

Assume that F_1 and F_3 are distributed respectively in CSP_1 and CSP_3 which will try together to link tuples stored in the two fragments by correlating the history of user queries, their execution time and their respective responses. In our example, CSP_1 will disclose that a client has executed $Q2_1$ to retrieve the tuple $\langle N.Baines, E_k(doc\_2)\rangle$ at the time t, while CSP_3 will disclose that the same client has executed $Q2_2$ to retrieve $\langle D.Annli\rangle$ at the time $t + n$. Using this information, CSP_1 might be able to infer that $E_k(doc\_2)$ is the encrypted value of 'doc_2'. Therefore CSP_1 can associate all patients having $Id\_doc =$ '$E_k(doc\_2)$' to the doctor whose name is 'N.Baines'.

To overcome this problem, the *Client* should have the ability to execute *semi join* queries and retrieve data from a fragment without the CSP (which stores the fragment) learning any information about the *semi join* condition values. To meet this requirement, we use a Private Information Retrieval keyword-based technique. PIR keyword-based was presented in [Chor et al. 1998] to retrieve data with PIR using keywords search over many data structures such as binary trees and perfect hashing. In the next section of this chapter, we will explain how we can use PIR keyword-based technique to ensure our *semi join* queries privacy requirement.

3.5.1 PIR System design

In the *Client* of our architecture, we give to *Query Executor* the ability to communicate with different Cloud providers through the PIR keyword-based protocol. In the *Server*, we add on each CSP a *PIR Server* as a front-end entity to answer

Query Executor's PIR queries. An adversary (a Cloud provider administrator) who can observe *Query Executor*'s PIR-encoded queries is unable to find out the clear content of the queries. Enforcing integrity on the *PIR server* side is straightforward since we assume that PIR servers will not attempt to wrongly answer *Query Executor*'s PIR queries.

The main purpose for using PIR keyword-based is to ensure the privacy of *semi join* queries. In our approach, this kind of queries is mainly executed over primary or foreign key attributes. In all existing PIR schemes, a common assumption is that the client should know the address of the block or the item to be retrieved. To satisfy this assumption in our approach, the *PIR server* will create an indexed structure over each indexed attributes in the database (e.g., attributes representing primary or foreign keys of the database relations). Therefore, we implement over these attributes two types of indices: B+ trees [Aho et al. 1983] and hash tables [Amble and Knuth 1974]. In the following subsections, we present then discuss the PIR keyword-based protocol using both index structures.

PIR based on B+ Trees

Private Block Retrieval (PBR) is a practical extension of PIR in which a user retrieves an n-bit block instead of retrieving only a single bit. Therefore, to be able to use B+ tree structure with the PBR, we consider each node or leaf in the B+ trees as a data block. However, in most cases, B+ tree nodes and leaves do not have the same size, so they cannot be used directly as all PBR approaches require that data blocks are of equal size. Thus, a required stage consists in adding padding data to nodes and leaves in order to have the same size for all B+ tree elements.

Using the PIR keyword-based query requires a setup phase in which the *Query Executor* and the *PIR server* exchange information. This setup phase is divided into two steps:

1. The *Query Executor* sends to the corresponding *PIR server* the Relation schema name and the attribute name over which the *semi join* query has to be performed.
2. When receiving the Relation schema name and the attribute name, the *PIR server* selects the corresponding B+ tree and sends its root node to the *Query Executor*.

After receiving the root node sent by the *PIR server*, the *Query Executor* will compare the list of keys contained in the root node with values used in the condition of the *Semi join* query in order to find out the indexes of the next nodes to be retrieved.

The *Query Executor* will subsequently perform PIR queries over chosen indexes to retrieve corresponding nodes. Once all items have been retrieved, the *Query Executor* combines them to build the result of the original *Semi join* query. Refer to Algorithm 3 and Algorithm 4 for a description of the PIR keyword-based protocol algorithms used in the *Server* and the *Client* parts. We illustrate the execution of a semi-Join query using the PIR keyword-based in the example below.

```
   input :
            BPT = {B_1, ..., B_n}   /* B-Plus Tree over indexed attributes*/
 1 while True do
 2     Request ← handle_client_request()
 3     if Request is PQR then
 4         /* PQR : Pre-Query Request */
 5         (TabName, AttriName) ← Request
 6         B ← GetAssociatedBPT(TabName, AttriName)
 7         Root_B ← GetRootNode(B)
 8         ReplyToClient(Root_B)
 9     end
10     if Request is PIRQ then
11         /* PIRQ : PIR Query */
12         result ← compute(Request)
13         ReplyToClient(result)
14     end
15 end
```

Algorithm 3: SemiJoin PIR keyword-based query (server)

Example 7. *Consider the query $Q2_2$ used in Example 4. We suppose that $\delta = \{doc\_3, doc\_69\}$. The execution of $Q2_2$ using PIR keyword-based protocol over B+ trees data structures is as follows:*

1. *The Query Executor sends $(Doctor, Id\_doctor)$ to the PIR server.*
2. *The PIR server sends the root node of the B+ tree corresponding to the received $(Doctor, Id\_doctor)$. Suppose that this root node is as presented in the table below.*

i_1	doc_11	i_2	doc_5	i_3
•		•		•

Note that i_1, i_2, i_3 are the indexes to the next level nodes. doc_11 and doc_5 are the root node keys.

```
input :
        tabName, attrName   /* Table and Attribute where the semi-join will be
performed */
        value               /* Semi-join condition value */
Node ← send_PQR_request(tabName, attrName)
while Node is not leaf_node do
    foreach elem in Node do
        findLink ← False
        if Key(elem) < value then
            Node ← PIR_Query(IndexOfLeftChild(elem))
            findLink ← True
            break
        end
    endfch
    if findLink = False then
        Node ← PIR_Query(IndexOfRightChild(elem))
    end
end
foreach elem in Node do
    if Key(elem) = value then
        return Data(elem)
    end
endfch
```

Algorithm 4: SemiJoin PIR keyword-based query (client)

3. *The Query Executor compares the elements of δ with the received nodes keys.*

   (a) *The Query Executor wants to retrieve the node containing the key doc_3, due to the fact that doc_11 < doc_3 < doc_5 (string comparison) and based on the received root node, the Query Executor will retrieve the node indexed by i_2.*

   (b) *The Query Executor needs also to retrieve the node containing the key doc_19, seeing that doc_5 < doc_69 (string comparison) and based on the received root node, the Query Executor will retrieve the node indexed by i_3.*

   For each index to be retrieved i_i, the Query Executor sends an encoded PIR query $PIR(i_i)$ to the PIR server. This process will be executed until the leaves of the B+ tree are reached. From the retrieved leaves, the Query Executor gathers tuples in which their keys are element of $\{doc\_3, doc\_69\}$.

Theorem 2. *Let $\mathcal{D}$ be a multi-relation normalized database, $\mathcal{F} = \{F_1, F_2\}$ be a fragmentation of $\mathcal{D}$, and Q be a multi-fragment query that joins records from both fragments F_1 and F_2. Consider that CSPs in which the fragments F_1 and F_2 are stored can collude to link data partitioned in these fragments, and that Q is evaluated using semi join operations. Sensitive relationships between F_1 records and F_2 records remain protected if and only if the privacy of the semi join sub-queries is guaranteed.*

Proof. To prove the Theorem 2, we use the following two sketches. The fist sketch proves that without ensuring *semi join* sub-queries privacy, collaborative CSPs can, in some cases, break data unlinkability, while the second sketch proves that, under a collaborative Cloud storage service providers model, protecting data unlinkability can only be guaranteed with the protection of the privacy of the *semi join* sub-queries.

SKETCH without using the PIR keyword-based protocol: Suppose that the *Cient* wants to execute a query which joins records from two fragments F_1 and F_2. Let us consider that the sub-query Q_1 executed over the fragment F_1 has returned n tuples. And the semi-join query Q_2 executed over F_2 has returned m tuples. Therefore, if CSPs that store F_1 and F_2 collude together to link tuples from Q_1 and Q_2 results, the probability to guess correctly the relationship between tuples (denoted using $\leftrightsquigarrow$) is:

$$Pr[Result(Q_1) \leftrightsquigarrow Result(Q_2)] = \frac{1}{m \times n}$$

Clearly, if m and n are small, CSPs will have a great chance to break data unlinkability.

SKETCH using the PIR keyword-based protocol: Let us consider that the *Client* attempts to perform a query which joins records from two fragments F_1 and F_2. According to our defined PIR keyword-based protocol, the *Client* will execute Q_1 over the fragment F_1 without using the keyword-based protocol. Next, the *Client* will send the table name T and the attribute name a on which the semi-join will be performed, the *Server* replies with the root node of the corresponding B+ tree. It is clear from the previous step that the CSP which stores F_2 can only know the attribute name and the table name on which the semi-join will be performed. After receiving the root node, the *Client* will use the PIR protocol to retrieve internal corresponding nodes until the leaves of the B+ tree are reached. The PIR protocol will ensure that the server will not know which nodes were retrieved by the *Client*. Moreover, all tuples are stored in the leaf level of the B+ tree. Therefore, in order to retrieve each record, the *Client* shall execute the same number of PIR queries. Rightfully, the only revealed information when using the PIR keyword-based protocol is the table name and the attribute name on which the semi-join has been performed. Therefore, if CSPs storing F_1 and F_2 collude together to break data unlinkability, they will be able only to infer

that the relation T_1 in F_1 over which Q_1 has been executed is linked to the relation T through the attribute a. Due to the fact that the foreign key in T_1 referencing the attribute a in T is encrypted, linking records is not possible. □

The particularity of B+ tree structures is that data appears only in the leaves while internal nodes is mainly used to guide the search. B+ tree's leaves are linked together to simplify sequential data access which gives the ability to perform in an efficient manner cardinality queries and range queries. However, the use of B+ tree data structure presents two disadvantages : first, as we have shown in the previous example, the *Query Executor* will use PIR queries to run down the tree and privately retrieve blocks (leaves) which contain records that match the semi-join condition. In each layer of the B+ tree, *Query Executor* should send a PIR query to the *PIR server* to get the addresses of the next layer nodes to be retrieved. Thus, for a k-layers B+ tree, *Query Executor* needs at least k PIR queries to reach the leaf layer, which is expensive in terms of communication and execution time. Second, several records which will not be part of the final result of the *semi join* query will be retrieved as a B+ tree leaf may contain several records having different index values.

PIR based on Hash Table

Hash table is a data structure that implements a mapping form keys to values. It is represented by an array in which data is accessed through a special index. The idea behind using hash tables is to map the indexed attribute (Primary key or foreign key) values to the set of corresponding records. These indexed attribute values will be the keywords which are used to search corresponding records stored in the hash tables. Hash tables are composed of set of sequential hash buckets. We will consider each set of records having the same keyword (indexed attribute value) as an hash buckets. The index of each bucket in the hash table is calculated using a minimal perfect hash function [Botelho et al. 2007] that maps n keywords to n consecutive integers.

We describe the use of hash table with the protocol PIR to perform *semi join* queries with the following four steps :

- `Setup step` – The *PIR server* create an hash table over each indexed attributes of the database. This stage is carried out only once as created hash tables will be used for subsequent *semi join* queries.
- `Step 1` – The *Query Executor* sends to the *PIR server* the name of the table and the name of the attribute on which the *semi join* is performed.

- `Step 2` – *PIR server* picks up the hash tree corresponding to the received couple (table, attribute) on which the *semi join* is performed and sends back to the *Query Executor* a set of metadata allowing the construction of the minimal perfect hash function used for building the corresponding hash table.
- `Step 3` – Using the received metadata, the *Query Executor* derives the minimal perfect hash function and calculates for each constant value in the *semi join* condition, the corresponding bucket index in which records corresponding to that constant value are stored. These calculated indexes are also the blocks numbers in the hash table index on the server. Next, the *Query Executor* will use PIR query to retrieve data blocks having the calculated indexes.

The advantage behind using hash tables lies in the fact that only one PIR query is needed to retrieve a data bloc instead of n PIR query when using a B+ tree (n represents the height of the B+ tree). Moreover, using hash tables, retrieved blocks will contain only records that match the original query of the user.

Example 8. *Consider the query* $Q2_2$ *used in Example 4. We suppose that* $\delta = \{doc\_3, doc\_69\}$. *the execution of* $Q2_2$ *using PIR keyword-based protocol over Hash tables data structures is as follows:*

1. *The Query Executor sends* $(Doctor, Id\_doctor)$ *to the PIR server.*
2. *The PIR server sends to the Query Executor a set of metadata allowing the construction of the minimal perfect hash function* f *used for building the corresponding hash table.*
3. *Using received minimal perfect hash function* f*, the Query Executor computes for each element in* δ*, the corresponding block index in which records corresponding to that element are stored. Suppose that* $f(doc\_3) = i_1$ *and* $f(doc\_69) = i_2$*, the Query Executor sends to the PIR server* $PIR(Doctor, i_1)$ *and* $PIR(Doctor, i_2)$ *to privately retrieve blocks having indexes* i_1 *and* i_2.

3.6 Implementation and Evaluation

We have developed a prototype for our approach, it is composed of two main components: (1) A Client entity developed in C++. Using regular expression offered by the *boost* library [Maddock 2001], we give the ability to the client Entity to

transform received queries into a QEP. Further, the Client uses the Crypto++ library [Dai 1995] to perform different cryptographic operations. (2) The server entity that uses STX B+ Tree [Bingmann 2008] and CMPH (C Minimal Perfect Hash) [Botelho and Ziviani 2007, Botelho and Zivian 2007] libraries to build and manipulate B+ tree and hash tables structures used by the PIR protocol. To give PIR functionality to the Server entity, we have used Percy++ [Goldberg 2007a, Goldberg 2007b]. Finally, for relational database server we used MySQL [Oracle].

3.6.1 Experimental Design

For our benchmarking, we used the relational database schema D composed of three relations as follows:

Patient($\underline{Id\_patient}, Name, SSN, Dob, Gender, ZIP, Illness$)
Doctor($\underline{Id\_doctor}, Name, Specialty$)
Examination($\underline{Id\_examination}, date, medical\_report, Id\_doc*, Id\_pat*$)

Note that the attributes Id_doc* and Id_pat* are two foreign keys that reference respectively the primary key of the table $Patient$ ($Id\_patient$) and the primary key of the table $Examination$ ($Id\_examination$). In tables $Patient$, $Doctor$ and $Examination$, we inserted respectively 10^6, 10^3 and 10^5 records. Fragments schemes obtained from the application of our secure fragmentation algorithm are represented below:

F_1 : **Patient**($\underline{salt}, enc, Id\_patient, Name, Dob, Gender, ZIP$)
Doctor($\underline{Id\_doctor}, Name, Specialty$)
F_2 : **Examination**($\underline{Id\_examination}, date, medical\_report, E_k(Id\_doc)*, E_k(Id\_pat*)$)
F_3 : **Patient**($\underline{salt}, enc, Illness$)

As we have previously seen, our approach is based on vertical fragmentation, the fragments of the table patient which are stored in F_1 and F_3 will be also composed of 10^6 records. Each fragment F_1, F_2 and F_3 of the database $\mathcal{D}$ should be stored in a different Service Provider. In our experiments, we used three virtual machines, with each representing a Service Provider and running MySQL 5.5.31. All experimentations have been performed on an eight cores server (Intel(R) Xeon(R) CPU x5355, 2.66 GHz) with 12 GB of RAM and running Ubuntu Linux 10.04.

3.6.2 Evaluation

As we have seen in the section 3.4, two kinds of queries can be used in order to query distributed fragments: (1) Queries that can be executed over only one fragment. (2) Queries which require the interrogation of several fragments to be evaluated. To evaluate the efficiency of our approach, we tested for each kind of query, and according to the number of retrieved records, the time required to execute the query. To evaluate the query of the type (1), we used the following query `Exp_Q1` :

```
Exp_Q1 :  SELECT Name, SSN, Dob
          FROM patient
          WHERE Gender = 'Male'
```

According to the used Fragments schemes, `Exp_Q1` is executed over the fragment F_1. To be able to control the number of returned records, we used the clause SQL `LIMIT` (`LIMIT 0, number_of_record`). The Figure 3.3 shows the execution costs per number of retrieved records. Note that in all experiments the cost of data transfer between the Server and the Client is negligible because both parties are installed in the same experimentation server.

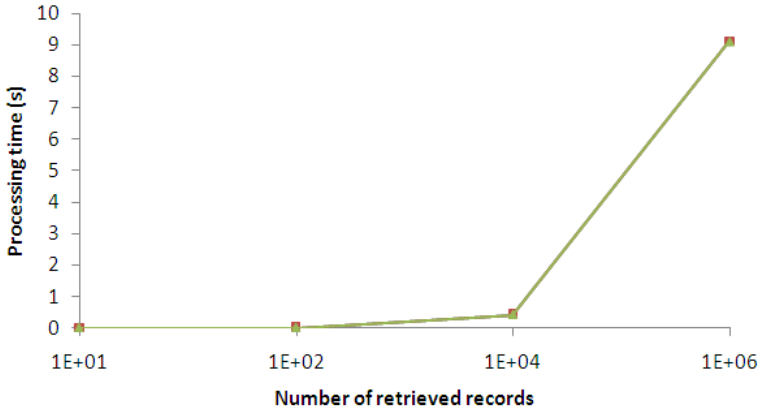


Figure 3.3 – Execution costs per number of retrieved records for the query Exp_Q1

For queries of the type (2), our approach will use *semi_join* with the PIR keyword-based protocol in order to join fragmented data while preserving the protection of sensitive associations. In this case, two kinds of data structure can be used : B+ tree and hash table.

For a better comparaison of the use of B+ trees and Hash tables with the keyword-based PIR, we executed the query `Exp_Q2` using the keyword-based PIR over both data structures.

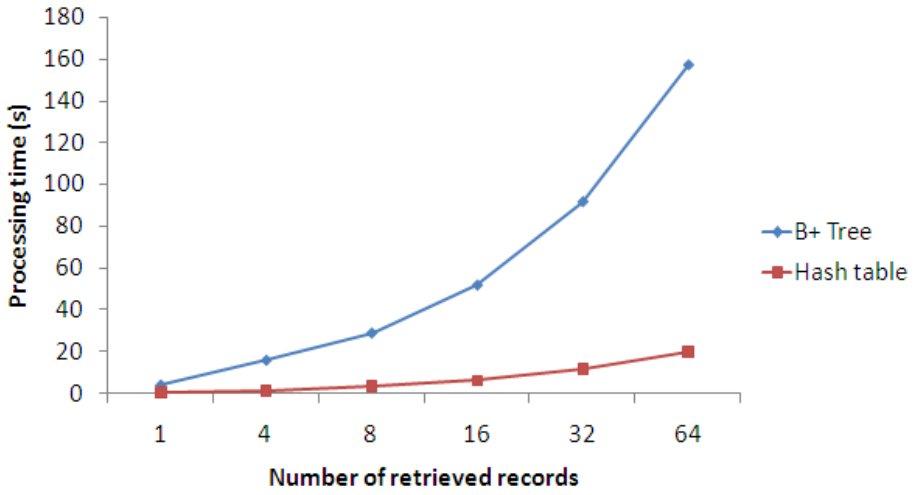


Figure 3.4 – Execution costs per number of retrieved records for the query Exp_Q1 over B+ Tree and Hash Table data structures

```
Exp_Q2 :  SELECT Name_pat, SSN
          FROM patient, examination e
          WHERE e.date= ?x
```

Figure 3.4 compares the processing costs for the query `Exp_Q1` using the keyword-based PIR over both data structures.

We make the following observations. First, as expected, the use of hash tables as the data structures used by the PIR keyword-based in order to perform *semi join* queries is much more efficient compared to the use of B+ trees with the PIR keyword-based. This can be explained by the fact that in the case of B+ tree, the *Client* must run down the tree using a PIR query in each level of the tree to retrieve data blocks stored in the leaf level of the B+ tree. While, in the case of hash tables, the *Client* needs to perform only one PIR query to get the data block containing requested records. For instance, the height of the corresponding B+ tree of the table *Patient* is 5, then the *Client* must perform 5 PIR queries to be able to retrieve corresponding records. Second, another reason to the inefficiency of the use of B+ trees compared to the use of hash tables with the PIR keyword-based is due to the fact that the size of constructed B+ trees are much more bigger than constructed hash tables which will introduce an execution overhead when performing PIR queries.

3.7 Conclusion and Contributions

In this chapter, we have presented an approach based on fragmentation, encryption and query privacy techniques enabling privacy-preserving of outsourced multi-relation databases. We presented different techniques that we have used to decompose multi-relational databases in the aim to protect sensitive associations, then we demonstrate how our decomposition techniques help in achieving data confidentiality. We presented a querying technique that optimizes and executes queries in this distributed system and how we can improve the security of the querying technique in order to protect data confidentiality under a collaborative Cloud storage service providers model.

Despite that our querying approach allows us to effectively and securely construct sensitive relations broken by the use of vertical fragmentation, it cannot efficiently execute queries over the attributes storing sensitive information, since the values assumed by those attributes are encrypted in such a way that makes performing operations (e.g., SUM, AVG, etc.) over them impractical. We strive to overcome this limitation in the next chapter by presenting an algorithm and tool set that determines an optimal balance between confidentiality and functionality of the sensitive outsourced data. That is, to use the functionalities (i.e. SUM, AVG, etc.) required by the queries to be executed over the outsourced database, we aim to get the different encryption schemes that can provide them when ensuring the best level of confidentiality.

CHAPTER 4

Combining Encryption-based Mechanisms to ensure Outsourced Data Confidentiality

4.1 Introduction

Encryption schemes have been proposed recently that allow to execute particular query operators over encrypted data and recent work by [Popa et al. 2011] shows that the general direct processing of encrypted data is an achievable goal, something recently confirmed in a larger industrial perspective [Grofig et al. 2014]. Following the idea of encrypting cleartext in so called "onions" allows to balance and match data processing functionality, i.e. each layer of an onion supports some SQL operations, with security, i.e. an onion structure introduces a total order with respect to the security properties of the chosen schemes. Yet, it is not practical to encrypt all columns in a table with the same onion structure. For example, columns may not require any encryption as they do not contain any sensitive material. Other columns may, for company specific compliance regulations, require to always be encrypted using a specific scheme when outsourced.

We believe that in order to further promote the wider industrial adoption of directly processing encrypted data, a more flexible configuration management is required before outsourcing the data from on-premise to a database-as-a-service cloud. In this chapter, we first present a policy-based configuration framework for encrypted data allowing the security administrator to specify the security policy to be applied over the outsourced data. Second, we propose an algorithm allowing to detect conflicts between security

and utility requirements. Third, we prove that selecting the optimal combination of encryption schemes that fit the defined policies with respect to the data owner's functional requirements (e.g. SQL that should be executed over the encrypted data) is NP-hard. Fourth, we therefore propose a heuristic, polynomial-time algorithm for finding a combination of encryption schemes that satisfies a policy P and provides the best security level.

The rest of this chapter is organized as follows, Section 4.2 describes the problem treated in this chapter. Section 4.3 presents the modeling of the used system and the modeling of the policy to be applied over the outsourced database. We show, for a given policy, how to detect the conflict between security and utility requirements involved in the policy and how to choose the combination of encryption schemes that enforces it. Section 4.4 presents a use case showing the application and the benefits of our approach in practice. Section 4.5 reports the implementation and evaluation of our approach. Finally, Section 4.6 reports our conclusions.

4.2 Problem Description

4.2.1 Adjustable Database Encryption

Encrypted databases can execute SQL queries over encrypted data. In this case, data is never decrypted inside the database server, but always remains encrypted. The key to the encryption and decryption functions solely resides at the client.

The main idea to processing queries in this way is property-preserving encryption. In property-preserving encryption a function $f(E(x), E(y))$ on ciphertexts $E(x)$, $E(y)$ returns the same result as $f(x, y)$. Hacigümüs et al. have described this concept for deterministic encryption and equality as a function [Hacigümüs et al. 2002]. They realized that many database operators, particularly selection and join, often use equality. Each data value is separately deterministically encrypted. Those database operators can then be used unmodified on encrypted data.

A limitation of the approach proposed in [Hacigümüs et al. 2002] was that inequality comparisons (range queries) were insufficiently supported. Agrawal et al. introduced order-preserving encryption [Agrawal et al. 2004]. Order-preserving encryption is property-preserving encryption for greater-than-or-equal comparisons. Using order-preserving encryption one can implement a large subset of SQL queries.

We have seen in Sections 2.2.1 and 2.2.3 that the security of order-preserving encryption and even deterministic encryption is still much debated. It is therefore better to choose the most secure encryption for a set of queries. If this set is unknown, then all data needs to be encrypted order-preservingly. Popa et al. presented a solution to this: adjustable (onion) encryption. Each data value is encrypted order-preservingly. This ciphertext is encrypted deterministically and the result is finally encrypted using standard randomized encryption secure against chosen plaintext attacks. Before a query is executed it is analyzed for the required encryption levels and the data values are adjusted (decrypted) to these levels. Hence, the most secure encryption can be chosen automatically.

4.2.2 Functional Requirements

As already mentioned the set of queries executed on the database pose a set of functional requirements. These requirements are captured as the functions executed on the ciphertext by the database operators.

In many cases a large subset of the queries to be executed is known. For example, when an application uses the database, one can analyse this application and extract the queries (maybe except for parameters). In many cases one can simply resort to the prepared SQL statements. If this subset of queries is known in advance, then it would be unwise to adjust the encryption during run-time. Although the adjustment process is performed only once, it can be quite costly. Each data value of an entire column needs to be decrypted which can sum to several MByte or even GByte of data.

Instead, the database can be encrypted to a "prepared" state and the adjustment process avoided. This leads to a significant shortening of the phase from a cold to a hot database. Real systems can go faster into production. Our approach is the first to support this analysis. We choose the appropriate encryption levels depending on the functional requirements of a set of queries.

4.2.3 Security Levels

The encryption levels of adjustable encryption correspond to different security levels. We claim that randomized encryption is at least as secure as deterministic encryption which is at least as secure as order-preserving encryption. We argue as follows.

Randomized encryption (RND) is semantically secure, i.e., it is secure against chosen plaintext attacks. We use AES in CBC for this encryption level. Clearly, then chosen plaintexts attacks are prevented.

Deterministic encryption (DET) allows chosen plaintext attacks, if the key is known or there is an encryption oracle. We only need symmetric encryption in encrypted database, such that it may be difficult to obtain the key or construct such an oracle. If a plaintext is encrypted and stored more than once, deterministic encryption also allows frequency attacks as in [Islam et al. 2012]. While not necessary, this may often – if not almost always – be the case in real databases. We therefore claim that deterministic encryption is less secure than randomized encryption. We use Pohlig-Hellman encryption, a symmetric key RSA variant, for this encryption level, in order to support proxy re-encryption [Kerschbaum et al. 2013b].

Order-preserving encryption (OPE) is also deterministic, such that all attacks on deterministic encryption also work for order-preserving encryption. In addition, it preserves the order, which may enable many more attacks. It was concluded that order-preserving encryption leaks at least half of the plaintext bits [Xiao and Yen 2012]. Clearly, order-preserving encryption is the least secure choice. We use the scheme by Boldyreva et al. [Boldyreva et al. 2009, Boldyreva et al. 2011] for this encryption level, which has been proven to be the optimally secure, immutable, order-preserving encryption scheme.

Next to these encryption levels we use homomorphic encryption (HE) for aggregation. Specifically, we use Paillier encryption [Paillier 1999]. Homomorphic encryption is secure against chosen plaintext attacks as is randomized encryption. Since for processing queries both ciphertexts need to be offered in parallel, they can be safely assumed to provide the same security level. Furthermore, similar to onion encryption, homomorphic encryption can be downgraded to deterministic encryption. As in the approach by Bellare et al.[Bellare et al. 2007], we can choose a deterministic randomization parameter. For downgrading we can simply select one ciphertext among the set of identical plaintexts. This has the added benefit that dictionary compression is as effective as on plaintext data [Kerschbaum et al. 2013a].

4.2.4 Security Requirements and The Need for Policy Configuration

Considering the security levels from Section 4.2.3 The data owner may realize that certain queries may put his data at risk. These queries may adapt the encryption level

to an unsafe state, e.g. order-preserving encryption, for a certain set of data. Even certain security standards, such as PCI-DSS [PCIDSS], may require certain encryption levels. Therefore the data owner may want to set certain policies on which encryption levels are allowed. He may want to prevent specific data from ever reaching a specific encryption state. For this, he needs the approach we propose in this chapter.

4.2.5 Policy Enforcement

The specified policies need to be enforced in the encrypted database. There is a crucial insight that enables prevention of certain encryption levels. If an encryption level is not present, it cannot be decrypted to. And vice versa, if an encryption should not be decrypted to, it does not need to be present. We therefore omit the encryption levels prevented by our policy. If one should not be able to decrypt to order-preserving encryption, the data value will not be encrypted order-preservingly. This has the positive side effect that ciphertexts may get smaller and encryption is more efficient.

The question remains what to do with queries that functionally require an encryption level that is prohibited by the security policy. In this case one ships the ciphertexts to the client, decrypts and executes the query on the client. The client query analysis algorithm of Kerschbaum et al. based on relational algebra, allows splitting a query into a local and a remote part [Kerschbaum et al. 2013c]. This way only the minimally necessary part of the query according to the security policy will be executed on the client.

4.3 Policy Configuration

In this section, we firstly present the modeling of the system and the specification of the policy. Afterwards, we present an algorithm allowing to detect conflicts between the constraints of the policy. We then propose an efficient algorithm allowing to enforce the policy while resolving the detected conflicts.

4.3.1 System Modeling

In our approach, data to be outsourced is stored in a relational database $\mathcal{D}$, which is composed of a collection of relational tables $\mathcal{T} = \{T_1, \cdots, T_n\}$, with each of these relational tables T_i containing a collection of attributes $\mathcal{A}_{T_i} = \{a_{1,i}, a_{2,i}, \cdots\}$. The system contains a toolbox $\mathcal{E}$ composed of a set of m encryption schemes $\{E_1, \cdots, E_m\}$

that can be used to protect outsourced data. Each encryption scheme $E_i \in \mathcal{E}$ is characterized by a security level l_i that provides and a set of functionalities $F_i \subseteq \mathcal{F}$ that satisfies. Let $\mathcal{F}$ be the set of functional requirements that can be required over the data to be outsourced and $\mathcal{L}$ be the set of security levels provided by $\mathcal{E}$.

4.3.2 Policy Modeling

We model, in a quite simple and powerful way, the requirements defined by the data owner. Those requirements are expressed through security and utility constraints. Security constraints are composed of confidentiality constraints and security threshold constraints.

Definition 7. ***(Confidentiality constraint)****. Given a relational table $T_i \in \mathcal{T}$ containing a list of attributes $\mathcal{A}_{T_i}$, a confidentiality constraint defined over T_i is a singleton set $CC = \{a\}$, where $a \in \mathcal{A}_{T_i}$.*

Semantically speaking, a confidentiality constraint CC states that the values assumed by the attribute in CC are considered sensitive and therefore must be protected.

Definition 8. ***(Security threshold constraint)*** *Given a relational table $T_i \in \mathcal{T}$ and an attribute $a \in \mathcal{A}_{T_i}$, a security threshold constraint TC_a over the attribute a is a security level l in $\mathcal{L}$. A security threshold constraint defined over the attribute a is well defined iff there exists a confidentiality constraint CC such that $a \in CC$.*

Security threshold constraints allow the data owner to specify a security level threshold for each sensitive attribute. The semantics of a security threshold constraint TC is that the security level of the sensitive attribute a must be at least as much secure as the security level l of TC.

Definition 9. ***(Utility constraint)*** *Given a relational table $T_i \in \mathcal{T}$ and an attribute $a \in \mathcal{A}_{T_i}$, an utility constraint UC_a over the attribute a is a set of functionality $F_a = \{f_1, \cdots, f_n\}$, where $F_a \subseteq \mathcal{F}$.*

Confidentiality protection is provided at the expense of data utility. A utility constraint offers the data owner the ability to require that some functionalities on his data must be provided, otherwise the data is useless.

4.3.3 Policy Conflict Detection

Policy conflicts occur when the objectives of two or more constraints cannot be simultaneously satisfied. Conflict detection aims at checking whether a set of constraints

contains conflicts. In our case, conflicts may occur between security constraints and utility constraints, more precisely, between security threshold constraints and utility constraints. To detect the conflicts, there are two steps. First, we must get for each security level $l \in \mathcal{L}$, the set of functionalities F_l which are satisfied by encryption schemes providing security levels that are at least as much secure as l. Then, for each sensitive attribute having $TC_a = l_a$ as a security threshold constraint and $UC_a = F_a$ as an utility constraint, we check if the set of functionalities F_{l_a} we got from the previous step for the level l_a is a superset of F_a, and if not, we deduce that there is a conflict between TC_a and UC_a. The set of conflicts in a defined policy are detected as described in Algorithm 5.

Example 9. *Let $\mathcal{L} = \{RND, DET, OPE\}$ be the set of security level that can be provided from the set of encryption schemes $\mathcal{E} = \{E_1, E_2, E_3\}$. Suppose that the E_1, E_2 and E_3 provide respectively RND, DET and OPE, and satisfy respectively the functionalities $\emptyset$, $\{Equality,\ Join\}$ and $\{Min,\ Max\}$. Suppose that we want to enforce a policy composed of two constraints $TC_a = DET$ and $UC_a = \{Join,\ Min\}$. By performing the first step of Algorithm 5, we deduce that $F_{RND} = \emptyset$, $F_{DET} = \{Equality,\ Join\}$ and $F_{OPE} = \{Equality,\ Join,\ Min,\ Max\}$. The second step of Algorithm 5 gives that $UC_a \nsubseteq F_{DET}$, which allows to deduce that TC_a and UC_a are conflicting constraints.*

4.3.4 Policy Satisfaction

The policy to be enforced over the outsourced database is composed of security and utility constraints. Those constraints can be satisfied through the application of encryption schemes. Our main challenge is to find for each sensitive attribute a in the outsourced database, the *best combination of encryption schemes* that can satisfy the set of security and utility constraints defined over a.

Definition 10. ***(combination of encryption schemes)*** *Let $\mathcal{E}$ be the set of available encryption schemes in the system, a combination of encryption schemes is a subset $C \subseteq \mathcal{E}$.*

Definition 11. *Let $C = \{E_1, \cdots, E_m\}$ be a combination of encryption schemes applied over the attribute a and l_i be the security level provided by the encryption scheme E_i, $1 \leq i \leq m$. The security level of the attribute a provided by the application of C is l, iff the following conditions hold:*

- $l \in \{l_1, \cdots, l_m\}$
- $\forall l_j \in \{l_1, \cdots, l_m\},\ l_j$ *is at least as secure as* l.

input :
$\mathcal{A}_s = \{a_1, \cdots, a_n\}$ /*sensitive attributes*/
$\mathcal{C}_t = \{TC_{a_1}, \cdots, TC_{a_n}\}$ /*security threshold constraints*/
$\mathcal{C}_u = \{UC_{a_1}, \cdots, UC_{a_n}\}$ /*utility constraints*/
$\mathcal{E} = \{E_1, \cdots, E_m\}$ /*encryption schemes*/
$\mathcal{L} = \{l_1, \cdots, l_p\}$ /*security levels*/
output:
$\mathcal{I}$ /*set of conflicts*/

```
1  Main
2  I = ∅
3  /* First step */
4  foreach l_i in L do
5      F_{l_i} = ∅
6      foreach E_j in E do
7          if (l_j is more secure or equal l_i) then
8              F_{l_i} = F_{l_i} ∪ F_j
9          end
10     endfch
11 endfch
12 /* Second step */
13 foreach a_k in A_s do
14     if ( UC_{a_k} ⊈ F_{TC_{a_k}} ) then
15         I = I ∪ {(a_k, UC_{a_k}, TC_{a_k})}
16     end
17 endfch
```

Algorithm 5: Conflict detection

Note that the previous definition requires the security level provided by the combination of schemes in C to be the lowest security level provided by the application of each encryption schemes in C. A strategy to find the combination of encryption schemes that satisfy the chosen policy consists of finding the *best combination of encryption schemes*, that is, it provides the highest level of protection for sensitive data, while minimizing the number of involved encryption schemes. We formalize this problem as follows:

Problem 1. ***(best combination of encryption schemes)*** *Let P be a policy, $\mathcal{C} = \{C_1, \cdots, C_n\}$ be a set of combinations of encryption schemes that satisfy the policy P, and l_i be the security level provided by the application of the combination C_i, with*

$1 \leq i \leq n$. *C_k is the best combination of encryption schemes in $\mathcal{C}$ that satisfy P iff the following conditions are satisfied:*

- $\forall C_j \in \mathcal{C}$, *$l_k$ is at least as secure as l_j.*
- $\forall C_j \in \mathcal{C},\ |C_k| \leq |C_j|$.

The problem of finding the best combination of encryption schemes is *NP-hard*. This is formally stated by the following theorem.

Theorem 3. *The problem of finding the best combination of encryption schemes is NP-hard.*

Proof. We prove the previous theorem by a reduction from the NP-hard problem of minimum hypergraph coloring [Garey and Johnson 1990], which is formulated as follows: *given a hypergraph $G(V, E)$, determine a minimum coloring of G, that is, assign to each vertex in V a color such that adjacent vertices have different colors, and the number of colors is minimized.*

We define the correspondence between finding the best combination of encryption schemes problem and the minimum hypergraph coloring problem as follows. Let a be a sensitive attribute, $TC_a = l$ be a security threshold constraint defined over a, $UC_a = \{f_{a_1}, \cdots, f_{a_n}\}$ be a utility constraint defined over a, and $\mathcal{E}_l = \{E_1, \cdots, E_m\}$ the set of encryption schemes that provide a security level which is at least as secure as l. Any vertex $v_i \in V$ corresponds to a functionality $f_i \in \mathcal{F}$. We denote e_a the edge in G which connects $v_{a_1}, \cdots, v_{a_n}$, corresponds to the constraint UC_a. The combination of encryption schemes $C = \{E_{i_1}, \cdots, E_{i_p}\}$, where $C \subseteq \mathcal{E}$ and each $E_{i_j} \in C$ satisfies the set of functionalities $F_j = \{f_{j,1}, \cdots, f_{j,k_j}\}$, satisfies the constraint UC_a correspond to a solution S for the corresponding hypergraph coloring problem. More precisely, S uses p colors. Vertices $\{v_{1,1}, \cdots, v_{1,k_1}\}$ corresponding to the functionality satisfied by E_{i_1} are colored using the first color, vertices $\{v_{q,1}, \cdots, v_{q,k_q}\}$ corresponding to the functionality satisfied by E_{i_q} are colored using the q-th color, and vertices $\{v_{p,1}, \cdots, v_{p,k_p}\}$ corresponding to the functionality satisfied by E_{i_p} are colored using the p-th color. Therefore, any algorithm finding the combination of encryption schemes that involved the minimal number of encryption mechanism while satisfying the constraint UC_a can be used to solve the minimum hypergraph coloring problem. □

Since the problem of finding the best combination of encryption schemes that satisfy a policy P is NP-hard, we cannot expect to be able to solve an instance of arbitrary size of this problem to optimality. Thus, heuristic resolution strategies are widely exploited to solve such a problem with a reasonable computational effort.

4.3.5 Heuristic Search

We propose a near-optimal heuristic for finding a combination of encryption schemes that satisfy a policy P. Our heuristic is based on a constructive method consisting of building a solution to the problem step by step from scratch. The used constructive method is based on choosing for each iteration, the *best satisfier* of the chosen policy.

Definition 12. ***(best satisfier)*** *Let $\mathcal{P}$ be a policy composed of two constraints: a security threshold constraint $TC_a = l$ and an utility constraint $UC_a = \{f_{a_1}, \cdots, f_{a_n}\}$. Both constraints are defined over the sensitive attribute a. Let $\mathcal{E} = \{E_1, \cdots, E_m\}$ be the set of available encryption schemes. $E_i \in \mathcal{E}$ is a best satisfier if the following conditions are satisfied:*

- *The security level l_{E_i} is at least as secure as l.*
- *$\forall E_j \in \mathcal{E}$, l_{E_j} is at least as secure as l and $|F_{E_i} \cap UC_a| \geq |F_{E_j} \cap UC_a|$, where F_E are the set of functionalities satisfied by E.*

The second condition in the previous definition states that E_i is the *best satisfier* if it satisfies the highest number of functionalities in UC_a compared to other encryption schemes in $\mathcal{E}$ that satisfy TC_a. Algorithm 6 shows our heuristic algorithm for computing for each sensitive attribute, a combination of encryption schemes that satisfy the constraints defined over it.

The algorithm takes as input the set of attributes $\mathcal{A}$ in the database to be outsourced, the policy $\mathcal{P}$ to be enforced over the set of attributes $\mathcal{A}$, the set of available encryption schemes $\mathcal{E}$ that can be used to enforce the policy $\mathcal{P}$, the set of security levels $\mathcal{L}$, and returns as output the set of combinations of mechanisms $\mathcal{S}$ that efficiently enforce the policy $\mathcal{P}$.

For conflicting constraints, the algorithm returns a set of propositions $\mathcal{CP}$ to aid in resolving the conflicts. The algorithm first initializes $\mathcal{S}$, $\mathcal{CP}$, $\mathcal{A}_s$ to the empty set and execute the procedure $get\_conflicting\_constraints$ which takes as parameters $\mathcal{P}$, $\mathcal{E}$, $\mathcal{L}$, and return the set of conflicts in the policy. The $get\_conflicting\_constraints$ procedure is represented by the Algorithm 5. Based on the confidentiality constraints in $\mathcal{P}$, the algorithm performs the first **foreach** loop to get all sensitive attributes $\mathcal{A}_s$. Then, for each sensitive attribute a_i having an unconflicting constraint it tries to get the best combination of schemes in terms of the provided security level. In order to meet the previous goal, we use the **while** loop to run down the set of security levels in $\mathcal{L}$ which are at least as secure as ($\geq_s$) TC_i starting from the highest one. For each security level sec_lev, we get from $\mathcal{E}$ the set $\mathcal{E}_{sec\_lev}$ of encryption schemes that provide security

```
input : A = {a_1, ··· , a_n}    /*database attributes*/
        P = {CC_1, ··· , CC_l, TC_1, ··· , TC_l, UC_1, ··· , UC_l}
        E = {E_1, ··· , E_m} /*encryption schemes*/
        L = {l_1, ··· , l_p} /*security levels*/
output: S    /*Solution*/
        CP   /*Conflict resolution propositions*/
1  Main
2  S = ∅, CP = ∅, A_s = ∅
3  Conflicts = get_conflicting_constraints(P, E, L)
4  foreach CC in P do
5  |  A_s = A_s ∪ CC
6  endfch
7  foreach a_i in A_s do
8  |  if (not (a_i, TC_i, UC_i) in Conflicts) then
9  |  |  sec_lev = get_the_highest_sec_lev(L);  Sol = ∅
10 |  |  while sec_lev ≥_s TC_i do
11 |  |  |  Sol = ∅, E_sec_lev = ∅
12 |  |  |  foreach E in E do
13 |  |  |  |  if (l_E ≥_s sec_lev and F_E ∩ UC_i ≠ ∅) then
14 |  |  |  |  |  E_sec_lev = E_sec_lev ∪ E
15 |  |  |  |  end
16 |  |  |  endfch
17 |  |  |  UC_temp = UC_i
18 |  |  |  while (UC_temp ≠ ∅ and E_sec_lev ≠ ∅) do
19 |  |  |  |  E_bs = get_first_elem(E_sec_lev)
20 |  |  |  |  foreach E in E_sec_lev do
21 |  |  |  |  |  if (|F_E ∩ UC_i| ≥ |F_E_bs ∩ UC_i|) then
22 |  |  |  |  |  |  E_bs = E
23 |  |  |  |  |  end
24 |  |  |  |  endfch
25 |  |  |  |  Sol = Sol ∪ E_bs;  E_sec_lev = E_sec_lev \ {E_bs}
26 |  |  |  |  UC_temp = UC_temp \ (F_E_bs ∩ UC_temp)
27 |  |  |  end
28 |  |  |  if (UC_temp = ∅) then
29 |  |  |  |  break
30 |  |  |  end
31 |  |  |  if (E_sec_lev = ∅) then
32 |  |  |  |  sec_lev = get_next_best_level(sec_lev, L)
33 |  |  |  end
34 |  |  end
35 |  |  S = S ∪ {(a_i, Sol, sec_lev)}
36 |  end
37 endfch
```

```
1  foreach (a_i, TC_i, UC_i) in Conflicts do
2      Prop = ∅
3      sec_lev = TC_i
4      while sec_lev ≠ NULL do
5          Prop = ∅
6          E_sec_lev = ∅
7          foreach E in E do
8              if (l_E ≥_s sec_lev and F_E ∩ UC_i ≠ ∅) then
9                  E_sec_lev = E_sec_lev ∪ E
10             end
11         endfch
12         UC_temp = UC_i
13         while (UC_temp ≠ ∅ and E_sec_lev ≠ ∅) do
14             E_bs = get_first_elem(E_sec_lev)
15             foreach E in E_sec_lev do
16                 if (|F_E ∩ UC_i| ≥ |F_E_bs ∩ UC_i|) then
17                     E_bs = E
18                 end
19             endfch
20             Prop = Prop ∪ E_bs
21             E_sec_lev = E_sec_lev \ {E_bs}
22             UC_temp = UC_temp \ (F_E_bs ∩ UC_temp)
23         end
24         sat_func = UC_i \ UC_temp
25         CP = CP ∪ {(a_i, Prop, sat_func, sec_lev)}
26         if (UC_temp = ∅) then
27             break
28         end
29         if (E_sec_lev = ∅) then
30             sec_lev = get_next_best_level(sec_lev, L)
31         end
32     end
33 endfch
```

Algorithm 6: Policy satisfaction

levels which are at least as secure as *sec_lev* and which can satisfy functionalities in UC_i. Next, we copy the set of required functionalities UC_i to UC_{temp}, and at each iteration of the next **while** loop, we get the *best satisfier* E_{bs} from $\mathcal{E}_{sec\_lev}$ according to the Definition 12. E_{bs} will be next added to the combination *Sol*, removed from $\mathcal{E}_{sec\_lev}$, and the required functionalities satisfied by E_{bs} will be removed from UC_{temp}. This **while** loop is terminated if: (1) all required functionalities in UC_{temp} are satisfied, in this case the set *Sol* represents the combination allowing to satisfy the constraints

defined over the attribute a_i; or (2) $\mathcal{E}_{sec\_lev}$ is empty, which means that there is no combination that satisfies UC_i in the security level sec_lev.

For each attribute a_i having a conflicting constraint, using the third outermost **foreach** loop, the algorithm gives additional proposition allowing to avoid the conflict. To meet this goal, we use the first **while** loop in the third outermost **foreach** loop to run down the set of security levels in $\mathcal{L}$ starting from TC_i. We perform the same operation as in the previous outermost **foreach** loop, except, for each sec_lev, we will add to the set of propositions $\mathcal{CP}$ the entry $(a_i, Prop, sat\_func, sec\_lev)$ stating that in the security level sec_lev, the combination of schemes $Prop$ is able to satisfy the set of functionalities sat_func required for the attribute a_i. These propositions may help the security administrator (data owner) to choose, from his point of view, the best trade off between security and utility.

Theorem 4. ***(Complexity)***. *Given a set of p attributes $\mathcal{A}$, a policy $\mathcal{P}$ composed of n confidentiality constraints, n security threshold constraints, n utility constraints, a set of m encryption schemes $\mathcal{E}$, and a set of r security levels, the complexity of the policy satisfaction algorithm (Algorithm 6) is $O(m^2 \cdot n \cdot r + r \cdot m + 2n)$.*

Proof. **(sketch)**. We suppose that we have p attributes having unconflicting constraints and q attributes having conflicting constraints, with $p + q = n$. According to Algorithm 5, the execution of the function $get\_conflicting\_constraints$ costs $O(r \cdot m + n)$. In Algorithm 6, the first *foreach* loop costs $O(n)$, the second *foreach* loop costs in the worst case $O(p \cdot r \cdot m^2)$, and the third *foreach* loop costs in the worst case $O(q \cdot r \cdot m^2)$. Finally, the overall time complexity of the Algorithm 6 is $O(m^2 \cdot n \cdot r + r \cdot m + 2n)$. □

4.4 Use Case

In this section, we present the use case. For our case study, we use a scenario based on the TPC-H [Doe] benchmark database. We first give an overview of the TPC-H benchmark database structure. Afterwards, we present the set of encryption schemes that can be used in our scenario, a set of functionalities required for processing the data, and policies to be applied over the TPC-H database. Finally, we illustrate the use of our previously presented policy satisfaction algorithm to enforce the chosen policy over the TPC-H database.

TPC-H Database

The TPC-H database is composed of 8 tables. Each attribute in TPC-H tables represents data for industrial resource management. TPC-H provides 22 queries consisting of different kind of SQL operations such as select, join, order by, etc. Figure 4.1 represents the conceptual model of the TPC-H database which includes foreign key relationships.

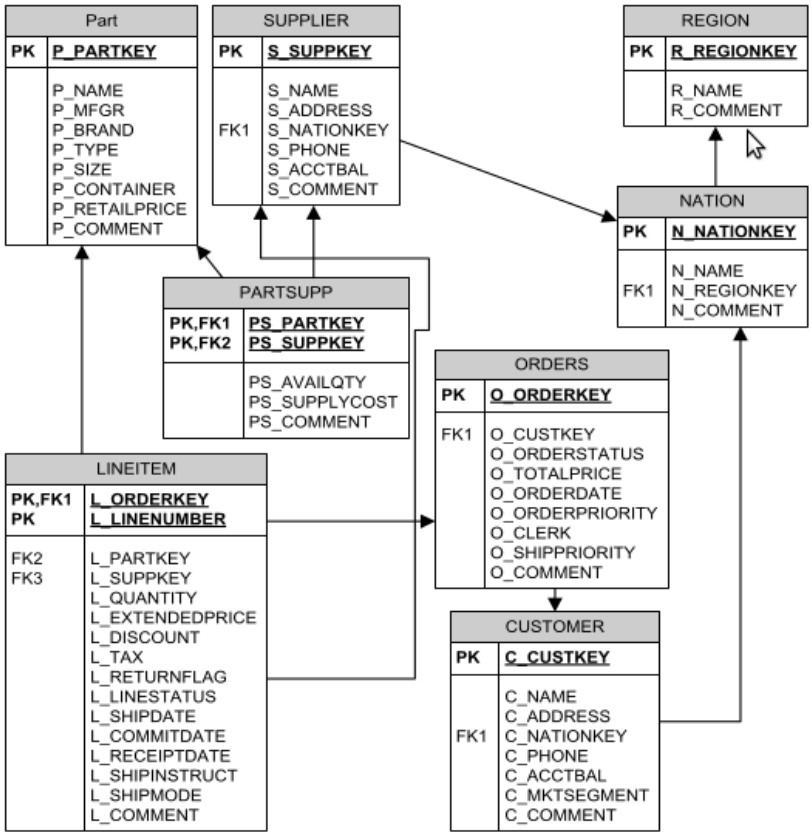


Figure 4.1 – TPCH database

System Design

As described in Section 4.3.1, the used system is composed of a relational database $\mathcal{D}$, a set of security layers $\mathcal{L}$, a set of functional requirements $\mathcal{F}$, and a toolbox $\mathcal{E}$. In our case study, $\mathcal{D}$ represents the TPC-H benchmark database, $\mathcal{L}$ will be composed of three security layers as explained Section in 4.2.3: RND (random layer), DET (deterministic layer) and OPE (order preserving layer). As we work with relational databases, the set of utility requirements are composed of some SQL operators that can be used to query the database. In addition, we define the functionalities *computation* representing

the numeric computation over the attributes (e.g., SET ATTR = ATTR + 30), and *order search* represeting the SQL operators ($>, \geq, <, \leq, between, min/max, order\ by$). Thus $\mathcal{F} = \{equality, join, group\ by, average, sum, computation, like, order\ search\}$. The toolbox $\mathcal{E}$ is composed of the following encryption schemes. For each encryption scheme, we extract and specify the provided security level and the set of satisfied functionalities as presented in Section 4.3.1.

AES-CBC. When used in CBC chaining mode, AES provides a probabilistic encryption which is semantically secure. Thus, it provides the security level *RND*. Despite that this encryption scheme does not leak any information about the plaintext values, it does not allow any efficient computation over encrypted data. Therefore, $l_{AES} = RND$ and $F_{AES} = \emptyset$.

Paillier [Paillier 1999]. It is based on secure probabilistic encryption which enables to perform computation over encrypted data. A Paillier cryptosystem provides indistinguishability under an adaptive chosen-plaintext attack (IND-CPA). It provides the security level *RND* and allows to perform *sum*, *avg* operations over the encrypted data. Thus, $l_{Plr} = RND$ and $F_{plr} = \{sum, avg, computation\}$.

SSE [Song et al. 2000]. SSE is a symmetric searchable encryption which is semantically secure (as long as there is no search token). It allows to perform search over encrypted data which gives the ability to perform MySQL's *like* operator. Based on these properties, the SSE can be specified by $l_{SSE} = RND$ and $F_{SSE} = \{like\}$.

Pohlig-Hellman. This is a deterministic encryption scheme allowing logarithmic time equality checks over ciphertexts. Pohlig-Hellman encryption cannot achieve the classical notions of security of probabilistic encryption because it leaks which encrypted values correspond to the same plaintext value. It provides the security level *DET* and allows to perform *equality*, *join*, and *group by* over the encrypted data. Thus, $l_{PH} = DET$ and $F_{PH} = \{equality, join, group\ by\}$.

Boldyreva [Boldyreva et al. 2009, Boldyreva et al. 2011]. Boldyreva propose an order-preserving, deterministic encryption which allows performing order operations over encrypted data. As mentioned in Section 4.2.3, in addition to the information leaked by having the deterministic property, it reveals the order between encrypted values. The encryption scheme provides the security level *OPE* and allows to perform *equality*, *join*, *group by*, and *order search* operations. Thus, $l_{Bdv} = OPE$ and $F_{Bdv} = \{equality, join, group\ by, order\ search\}$.

The Policy

In our scenario a security administrator (data owner) of the TPC-H benchmark database requires that the following security rules must be enforced:

- **Rule 1.** The given discount for any Order should always remain *top secret*.
- **Rule 2.** The account balance for a customer as well as our suppliers should always remain *top secret*.
- **Rule 3.** The Name and Address of our suppliers should be *confidential*.
- **Rule 4.** The supply cost of individual suppliers must be *confidential*.
- **Rule 5.** Any pricing information must remain *secret*.
- **Rule 6.** All other information in the database should be *unclassified*.

The security administrator used four levels to classify the data. The *top secret* classification levels means that any leaked information about the data will cause grave damage. The *secret* level means that some information about the data values can be leaked if they do not lead to reveal its values. The *confidential* level means that additional information about the data values can be leaked if they do not lead to reveal the values itselves. A *Unclassified* level implies that the data are not sensitive.

According to the properties of the security levels in $\mathcal{L}$ described in Section 4.2.3, we associate the *top secret* classification levels to the *RND* security level, the *secret* classification level to the *DET* security level, and the *confidential* classification level to the *OPE* security level. The previous rules are specified as follows:
Rule 1. It involves the attribute *L_DISCOUNT* of the table *LINEITEM*. This rule is specified using the folowing confidentiality and security threshold constraints:

- $CC_1 = \{L\_DISCOUNT\}$, $TC_1 = RND$.

Rule 2. This rule involves the attributes *C_ACCTBAL* and *S_ACCTBAL* from the tables *CUSTOMER* and *SUPPLIER*. It is specified using the following constraints:

- $CC_2 = \{C\_ACCTBAL\}$, $TC_2 = RND$.
- $CC_3 = \{S\_ACCTBAL\}$, $TC_3 = RND$.

Rule 3. It involves the attributes *S_NAME*, *S_ADDRESS*, and *S_NATIONKEY* from the table *SUPPLIER*. It is specified using the following constraints:

- $CC_4 = \{S\_NAME\}$, $TC_4 = OPE$.

- $CC_5 = \{S\_ADDRESS\}$, $TC_5 = OPE$.
- $CC_6 = \{S\_NATIONKEY\}$, $TC_6 = OPE$.

Rule 4. It involves the attribute *PS_SUPPLYCOST* from the table *SUPPLYCOST*. This rule is specified using the following constraints:

- $CC_7 = \{PS\_SUPPLYCOST\}$, $TC_7 = OPE$.

Rule 5. This rule involves the attributes *P_RETAILPRICE*, *L_EXTENDEDPRICE* and *O_TOTALPRICE* from tables *PART*, *LINEITEM* and *ORDERS*. It is specified using the following constraints:

- $CC_8 = \{P\_RETAILPRICE\}$, $TC_8 = DET$
- $CC_9 = \{L\_EXTENDEDPRICE\}$, $TC_9 = DET$
- $CC_{10} = \{O\_TOTALPRICE\}$, $TC_{10} = DET$

There is no need to specify Rule 6 as the data assumed by the attributes concerned by this rule are not sensitive. That is, this data can be stored in plaintext.

The security administrator gives examples of queries which should be executed efficiently over the TPC-H database. From these set of queries, we extract only the queries involving sensitive attributes described in the policy, which are illustrated in Figure 4.2. These queries enable us to extract the set of functionalities required for each sensitive attribute in the TPC-H database. Table 4.1 shows, for each sensitive attribute, the queries on which the attribute is involved and the set of required functionalities. These functional requirements are specified using the following utility constraints:

- $UC_1 = \{computation, sum, order\ search\}$
- $UC_2 = \{group\ by, sum\}$
- $UC_3 = \{order\ search\}$
- $UC_4 = \{order\ search, group\ by\}$
- $UC_5 = \{like\}$
- $UC_6 = \{join\}$
- $UC_7 = \{equality\}$
- $UC_8 = \emptyset$
- $UC_9 = \{sum, computation\}$
- $UC_{10} = \{group\ by, order\ search\}$

Q1:
SELECT L_RETURNFLAG, L_LINESTATUS,
SUM(L_QUANTITY) AS SUM_QTY,
SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
SUM(1-L_DISCOUNT) AS SUM_DISC_PRICE,
AVG(L_QUANTITY) AS AVG_QTY,
FROM LINEITEM
WHERE
L_SHIPDATE <= '2010-01-15'
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG,L_LINESTATUS

Q2:
SELECT S_ACCTBAL, S_NAME, N_NAME, P_PARTKEY, P_MFGR, S_ADDRESS, S_PHONE, S_COMMENT
FROM PART, SUPPLIER, PARTSUPP, NATION, REGION **WHERE**
P_PARTKEY = PS_PARTKEY AND
S_NATIONKEY = N_NATIONKEY
PS_SUPPLYCOST = 1000
ORDER BY S_ACCTBAL **DESC**, N_NAME, S_NAME

Q3:
SELECT SUM(L_DISCOUNT) AS REVENUE
FROM LINEITEM
WHERE L_SHIPDATE >= '2010-01-01' AND
L_SHIPDATE < '2010-01-01'
AND L_DISCOUNT BETWEEN .06 - 0.01 AND .06 + 0.01 AND L_QUANTITY < 24

Q4:
SELECT N_NAME AS NATION,
L_EXTENDEDPRICE*(1-L_DISCOUNT) AS AMOUNT
FROM PART, SUPPLIER, LINEITEM, NATION
WHERE
S_SUPPKEY = L_SUPPKEY
AND S_NATIONKEY = N_NATIONKEY
AND S_ADDRESS LIKE '%%RENNES%%'
Group By N_NAME.

Q5:
SELECT TOP 20 C_NAME, C_ACCTBAL,
N_NAME, C_ADDRESS, C_PHONE, C_COMMENT
FROM CUSTOMER, ORDERS, LINEITEM, NATION
WHERE C_CUSTKEY = O_CUSTKEY AND
L_ORDERKEY = O_ORDERKEY AND
L_RETURNFLAG = 'R'
GROUP BY C_CUSTKEY, C_NAME, C_ACCTBAL, C_PHONE
ORDER BY C_NAME.

Q6:
SELECT C_NAME, O_ORDERDATE,
O_TOTALPRICE, SUM(L_QUANTITY)
FROM CUSTOMER, ORDERS, LINEITEM
WHERE C_CUSTKEY = O_CUSTKEY AND
O_ORDERKEY = L_ORDERKEY
GROUP BY C_NAME, C_CUSTKEY, O_TOTALPRICE
ORDER BY O_TOTALPRICE DESC.

Q7:
SELECT TOP 100 S_NAME, COUNT(*) AS NUMWAIT
FROM SUPPLIER, LINEITEM L1, ORDERS, NATION
WHERE S_SUPPKEY = L1.L_SUPPKEY AND
O_ORDERKEY = L1.L_ORDERKEY AND
L1.L_RECEIPTDATE> L1.L_COMMITDATE
GROUP BY S_NAME
ORDER BY NUMWAIT DESC, S_NAME.

Q8:
SELECT CNTRYCODE, COUNT(*) AS NUMCUST,
SUM(C_ACCTBAL) AS TOTACCTBAL
FROM
(**SELECT** SUBSTRING(C_PHONE,1,2) AS CNTRYCODE, C_ACCTBAL
FROM CUSTOMER
WHERE
SUBSTRING(C_PHONE,1,2) IN ('13', '31', '23', '29'))
GROUP BY CNTRYCODE

Figure 4.2 – Queries involving sensitive attributes

Policy Enforcement Results

Using the Algorithm 6, we get from the toolbox, for each sensitive attribute, the encryption scheme or the combination of encryption schemes that satisfies the policy. The results of the application of Algorithm 6 over our use case are the followings:

Sensitive attributes	Functionalities
L_DISCOUNT	*computation*(Q1,Q3,Q4) *sum*(Q1,Q3,Q4) *order search*(Q3)
C_ACCTBAL	*group by*(Q5) *sum*(Q8)
S_ACCTBAL	*order search*(Q2)
S_NAME	*order search*(Q2,Q7) *group by*(Q7)
S_ADDRESS	*like*(Q4)
S_NATIONKEY	*join*(Q2,Q4)
PS_SUPPLYCOST	*equality*(Q2)
P_RETAILPRICE	
L_EXTENDEDPRICE	*sum*(Q1,Q3) *computation*(Q3,Q4)
O_TOTALPRICE	*group by*(Q6) *order search*(Q6)

Table 4.1 – Required functionalities for sensitive attributes

1. C_ACCTBAL: conflict detected (TC_2 and UC_2)
   Conflicts resolution propositions:
   - $[Paillier]$ (RND), satisfied utility requirements: $\{sum\}$ (Q8)
   - $[Paillier, Pohlig - Hellman]$ (DET), satisfied utility requirements: $\{group\ by, sum\}$ (Q8, Q5)
2. L_EXTENDEDPRICE: $[Paillier]$ (RND), satisfied utility requirements: $\{sum, computation\}$ (Q1,Q3,Q4).
3. PS_SUPPLYCOST: $[Pohlig - Hellman]$ (DET), satisfied utility requirements: $\{equality\}$ (Q2).
4. L_DISCOUNT: conflict detected (TC_1 and UC_1)
   Conflicts resolution propositions:
   - $[Paillier]$ (RND), satisfied utility requirements: $\{sum, computation\}$ (Q1,Q4).
   - $[Paillier, Boldyreva]$ (OPE), satisfied utility requirements: $\{sum, order\ search, computation\}$ (Q1,Q3,Q4).

5. S_ADDRESS: [SSE] (RND), satisfied utility requirements: {$like$} (Q4).
6. S_NAME: [$Boldyreva$] (OPE), satisfied utility requirements: {$order\ search, group\ by$} (Q2,Q7).
7. S_NATIONKEY: [$Pohlig - Hellman$] (DET), satisfied utility requirements: {$join$} (Q2,Q4).
8. S_ACCTBAL: conflict detected (TC_3 and UC_3)
   Conflicts resolution propositions:
   - [$AES - CBC$] (RND), satisfied utility requirements: $\emptyset$.
   - [$Boldyreva$] (OPE), satisfied utility requirements: {$order\ search$} (Q2).
9. P_RETAILPRICE: [$AES - CBC$] (RND).
10. O_TOTALPRICE: conflict detected (TC_{10} and UC_{10})
    Conflicts resolution propositions:
    - [$Pohlig - Hellman$] (DET), satisfied utility requirements: {$group\ by$}.
    - [$Boldyreva$] (OPE), satisfied utility requirements: {$group\ by, order\ search$} (Q6).

Result 1 shows the satisfaction of the constraints defined over the attribute C_ACCTBAL. A conflict between the constraints TC_2 and UC_2 has been detected. Thus, our algorithm gives the data owner two propositions in order to resolve the conflict. The first proposition states that the data owner can preserve the RND security level through the application of the $Paillier$ encryption scheme, however only the *sum* functionality will be provided and therefore the query Q5 cannot be executed efficiently over the encrypted data. The second proposition gives the data owner the ability to decrease the required threshold security level to DET in order to allows the application of the combination [$Paillier, Pohlig - Hellman$] which satisfies the required utility constraints. Result 2 states that the encryption scheme $Paillier$ can be applied to enforce the set of security and utility requirements defined over the attribute L_EXTENDEDPRICE. Result 3, shows that security and utility constraints defined over the attribute PS_SUPPLYCOST can be enforced through the application of $Pohlig - Hellman$ encryption scheme. Result 4 shows that there is a conflict between the constraints TC_1 and UC_1 and proposes two solution to reconcile the conflict. Result 5 states that the encryption scheme SSE can be applied to enforce the set of security and utility requirements defined over the attribute S_ADDRESS. Result 6 shows that the set of security and utility constraints defined over the attribute

S_NAME can be enforced via the application of *Boldyreva* encryption scheme. Result 7 states that the encryption scheme $Pohlig-Hellman$, when applied, can enforce the of security and utility requirements defined over the attribute S_NATIONKEY. For this result, we remark that our algorithm has chosen the best encryption scheme in terms of provided security level, as the *Boldyreva* encryption scheme can also be used to enforce security and utility requirements defined over the attribute S_NATIONKEY. Result 8 shows the conflict detected between TC_3 and $UC3$ and proposes two solutions two overcome the conflict. Result 9 confirms that the application of $AES-CBC$ can enforce the constraints defined over the attribute P_RETAILPRICE. Finally, result 10 shows that there is a conflict between TC_10 and UC_10 and proposes two solution allowing to reconcile the conflict.

It is important to note that sequentially applying a combination of encryption schemes (e.g. [Paillier and Pohlig-Hellman] in one "onion") over an attribute may not provide the functionalities provided by each encryption scheme. This problem can be resolved by duplicating the values of the attribute over which the two mechanisms are to be applied and apply each mechanism separately.

4.5 Implementation and Evaluation

In this section, we evaluate the efficiency of our approach by using a web application as well as a large trace of SQL queries (For extracting the functional requirements for the used applications). Our prototype consists of a Java library composed of an SQL query parser, a database schema reader module (DRM), a policy specification module, and policy satisfaction module. The query parser is used to extract the utility requirements (or functionalities) that should be provided over each attribute in the target database. We use the open source JSqlParser [JSqlParser] as an SQL query parser. The DRM is used to connect and retrieve the schema of the target database to allow the user of our library specifying the policy to be applied. Our DRM implementation supports both Postgres 9.0 and MySQL 5.1 databases.

4.5.1 Experimental Design

We ran the all experiments on a server with Intel core i7 2.50 GHz, 16 GB of RAM, and running Ubuntu 14.04. We evaluate the effectiveness of our policy satisfaction algorithm using the web application phpBB. Table 4.2 reports the number of attributes in the relational database used by the phpBB web application as well as the number

of queries we have used to extract the functionalities that should be provided for the attributes in the used relational database.

Application	**Total Attr.**	**Total Num. of queries**
phpBB	563	11992

Table 4.2 – The web application used to evaluate our policy satisfaction approach

The encryption toolbox we used to enforce policies over the outsourced database is composed of 5 encryption mechanisms. Table 4.3 illustrates the functionalities as well as the security level provided by each of the used encryption mechanisms and their corresponding supported data types.

Encryption Scheme	**Data Type**	**Security Level**	**Provided Functionalities**			
			ES	**OS**	**CP**	**KS**
AES-CBC	All	RND	✗	✗	✗	✗
Paillier	Numeric	RND	✗	✗	✓	✗
SSE	Textual	RND	✗	✗	✗	✓
Pohlig-Hellman	All	DET	✓	✗	✗	✗
Boldyreva	Numeric	OPE	✓	✓	✗	✗

Table 4.3 – Encryption mechanisms used in evaluating our approach with their corresponding supported data types, ensured security levels, and provided functionalities over encrypted data. 'ES' represents the equality search functionality, 'OS' represents the order search functionality, 'CP' represents the computation requirements, and 'KS' is for keyword search functionality.

4.5.2 Evaluation

In order to show the performances of our approach, we examine the computation duration cost of our policy satisfaction approach over the relational database used by phpBB in function of the number of sensitive attributes, the number of threshold constraints, and the number of utility constraints. Figure 4.3 shows that the time needed to satisfy a policy increases, dependently from the number of sensitive attributes as well as the numbers of defined threshold constraints, and utility constraints. The measure confirms that a linear complexity in terms of the number of sensitive attributes and the numbers of defined threshold constraints and utility constraints is achieved, which is conform with the result proved in Theorem 4.

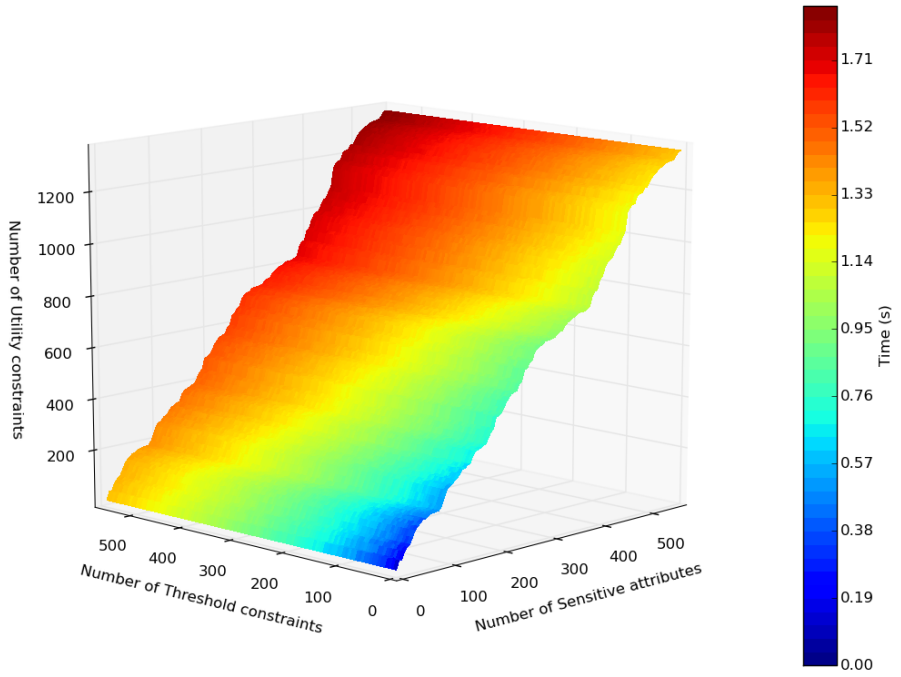


Figure 4.3 – Policy satisfaction evaluation in function of the number of sensitive attributes, the number of threshold constraints, and the number of utility constraints

4.6 Conclusion and Contributions

Searchable, yet encrypted databases appear to be one promising building block of a secure cloud offering. In order to help companies migrate data from on-premise to the cloud, tools are needed to help decide about the best acceptable trade-off between functionality and security requirements. In this chapter, we presented a set of algorithms which help to analyze functionality and security requirements when configuring an encrypted database following an onion-based approach. We reasoned about their formal characteristics as well as discussed their application in an enterprise use case on basis of the TPC-H benchmark. The assumption that data may be labelled as we proposed may appear oversimplified, but industrial experience shows that even in complex applications this is sufficient to cover the evaluation results of a typical 3x3 risk matrix.

The approach proposed in this chapter complements our contribution presented in the chapter 3 since it allows to overcome its limitation by allowing a data owner to get, for the attribute storing sensitive data, the combination of encryption schemes that

provides the best trade-off between the security and utility of the outsourced sensitive information.

The contribution presented in this chapter is the result of a collaboration with SAP AG (Karlsruhe, Germany). Our developed tool is integrated in the SAP's recent security research project SEEED [SAP].

Part II

Heterogeneous Security and Utility Requirements Specification and Enforcement over Outsourced Data

CHAPTER 5

An Epistemic Temporal Logic based language for Specifying Security Policy and Security Mechanisms

5.1 Introduction

In recent years, the concept of data outsourcing has become quite popular since it offers many features, including reduced costs from saving in storage, increasing availability as well as minimizing management effort. Many security-related research issues associated with data outsourcing have been studied focusing on data confidentiality [Hacigümüs et al. 2002, Bkakria et al. 2013b], data authentication and integrity [Mykletun et al. 2004, Narasimha and Tsudik 2005], Copyright protection [Sion 2008, Gross-Amblard 2011], privacy and anonymity [Sweeney 2002, Machanavajjhala et al. 2006, Li et al. 2007], because outsourced data often contains highly sensitive information which will be stored and managed by third parties. To tackle those traditional security issues, data protection mechanisms have recently been the focus of huge interest, especially cryptographic and information hiding techniques such as encryption, anonymization, watermarking, fragmentation, etc. These mechanisms are known to be efficient when used independently. However, in many situations they have to be combined to ensure security requirements.

In Cloud storage model, the goal of policies is to achieve a set of security and utility properties over the outsourced data. However, the relation between these properties and the existing security mechanisms defined to protect outsourced data is not always obvious and a question naturally emerges: how to bridge the gap in such a case?

The complexity of Cloud storage security policies is increasing due to the interplay of several heterogeneous security and efficiency requirements that should be harmonized and specified into consistent policies. An important related issue is the trade-off between the level of provided security and the target system usability, especially when it is open to the Internet (e.g., Cloud service Storage).

Logic-based languages are specifically appealing for security policy specification. One main benefit resides in their clean and straightforward semantics, suitable for specification and validation. In addition, logic-based languages can be designed to be expressive enough to specify all security properties that might be required over a target system. Their declarative nature provides a good compromise between simplicity and expressiveness.

Stemmed from the absence of relevant work in the area of formal policy specification and deployment for outsourced data, in this chapter, we present our third contribution [Bkakria et al. 2014a]. That is, we strive to design an expressive formal language allowing us to formally specify the data structure storing the information to be outsourced, formally specify, as finely as possible, the policy to be applied over the data to be outsourced, and formally specify the existing security mechanisms that can be used to protect the data to be outsourced.

The rest of this chapter is organized as follows. Section 5.2 introduces the formal language that will be used in our approach. Section 5.3 shows the data model formal specification. Section 5.4 formalizes the main objectives that data owners attempt to reach when using a Cloud storage model . Section 5.5 shows the modeling of the policy to be applied over the outsourced information. Section 5.6 presents the specification of the existing security mechanisms that can be used to satisfy a defined policy. Section 5.7 concludes the chapter.

5.2 An Epistemic Temporal Logic based language

In this section, we introduce the first-order temporal epistemic logic language $\mathcal{L}$ that we will use to formalize: (1) the data structure storing the information to be outsourced, (2) the policy to be applied over it, and (3) the security mechanisms that might be used to enforce the policy.

The first-order temporal epistemic logic language $\mathcal{L}$ can be seen as a composition of a first order logic (predicate logic), epistemic logic [Hintikka 1986], and temporal logic [Davis and McKim 1976]. The aims behind combining these tree formal systems are the following:

- Data structures traditionally used to store information are often complex structures composed of a set of associated data objects. First, order logic allows us to formally specify the data objects and their associations through the use of predicate symbols. Moreover, it provides expressive power through the availability of quantifiers and variables which allows us to state facts about data objects of the structure without enumerating the particular objects.

- The security requirements in the policy to be enforced are often defined over one or many particular moments of the life cycle of the outsourcing process (as we will later see in Section 5.5). Temporal logic provides a set of temporal operators allowing us to explain those security requirements over the life cycle of the outsourcing process.

- When dealing with security issues in the Cloud storage model, the ability to describe the knowledge of the involved entities (e.g., Cloud provider, external adversaries, authorized users, etc.) becomes particularly important. Epistemic logic provides operators allowing to specify informational aspects related to knowledge.

These three formal systems have the advantages of a well-defined semantics, an existing body of theoretical work related to, axiomatisations and complexity, see for example [Halpern and Vardi 1989], and sound and complete proof methods for example [Dixon and Fisher 2000].

5.2.1 Syntax

The first-order temporal epistemic logic language $\mathcal{L}$ is based on $KL_{(n)}$ [Dixon and Fisher 2000], which represents the fusion of first order linear-time temporal logic with multi-modal S5 [Lewis and Langford 1932]. The temporal modality is interpreted over a discrete linear model of time with infinite future and finite past, such a logic has been studied in detail [Halpern and Vardi 1989] and is the most commonly used temporal logic of knowledge. $\mathcal{L}$ is made up of a set of predicates $\mathcal{P}$, propositional connectives $\vee$, $\wedge$, $\neg$, $\rightarrow$ and $\leftrightarrow$, the quantifiers $\forall$, $\exists$. We take the usual set of future connectives $\Diamond$ (Sometime, or eventually) and $\Box$ (always) [Gabbay et al. 1980]. For knowledge we assume a set of agents $A_g = \{1, \cdots, m\}$ and use a set of unary modal connectives K_j, for $j \in A_g$, in which a formula $K_j\psi$ is to be read as "agent j knows ψ".

Definition 13. *Let P_i be a predicate of arity n in $\mathcal{P}$. The set of well-formed formulas of $\mathcal{L}$ is defined as follows:*

$$\phi ::= P_i(t_1, \cdots, t_n) \mid K_i\phi \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \Diamond\phi \mid \Box\phi \mid \phi \rightarrow \phi \mid \phi \leftrightarrow \phi \mid \exists x\phi \mid \forall x\phi \mid \phi \xrightarrow{refine} \phi$$

where $t_1, \cdots, t_n$ are terms (variables or constants).

The language $\mathcal{L}$ is extremely expressive. It can be used to specify complex systems, the temporal evolution of those systems, and the temporal evolution of agents' knowledge, as we will see later on in this chapter.

5.2.2 Semantics

The semantics we used for $\mathcal{L}$ are inspired by the ones proposed in [Dixon and Fisher 2000].

Definition 14. *Let States be the set of all possible states of the world.*

Definition 15. ***(Timeline.)*** *A timeline t is an infinite linear discrete sequence of states, indexed by natural numbers. Let Timelines be set of all timelines.*

Definition 16. ***(Interpretation.)*** *An interpretation $\mathcal{I}$ of the language $\mathcal{L}$ is the couple $\mathcal{I} = (States, \mathcal{S})$, where $\mathcal{S}$ represents a set of classical first-order structures. Each $I_s \in \mathcal{S}$ with a non-empty domain D_s assigns to a state $s \in States$, a predicate $I_s(P) : D_s^n \rightarrow \{True, False\}$ for each n-places predicate $P \in \mathcal{P}$.*

Definition 17. ***(Model.)*** *An model M for $\mathcal{L}$ is a structure $\langle TLs, \mathcal{R} = \{R_1, \cdots, R_n\}, \mathcal{I}, \Phi\rangle$, where:*

- *TLs is a set of timelines;*
- *$\forall i \in A_g$, $R_i \subseteq States \times States$ represents the accessibility relation of an agent over States;*
- *$\mathcal{I}$ is an interpretation;*
- *Φ is a transition function which defines transitions between states due to the application of mechanisms (actions). $\Phi(s, m_k) = s'$ if the mechanism m_k transits a model from a state s to state s'.*

The semantics of our language $\mathcal{L}$ are described through the definition of satisfaction relation $\models$.

Definition 18. *Given a model M for $\mathcal{L}$, a state s, the satisfaction relation $\models$ for a formula ψ of $\mathcal{L}$ is defined as follows:*

- $(M, s) \models P(t_1, \cdots, t_n) \iff I_s(P)(v(t_1), \cdots, v(t_n)) = True$, *where v is a valuation function that assigns, for each $t_i \in \{t_1, \cdots, t_n\}$ an element in D_s.*
- $(M, s) \models \neg\psi \iff (M, s) \not\models \psi$
- $(M, s) \models \psi \rightarrow \varphi \iff (M, s) \not\models \psi$ *or* $(M, s) \models \varphi$
- $(M, s) \models \psi \leftrightarrow \varphi \iff (M, s) \models (\psi \rightarrow \varphi) \wedge (\varphi \rightarrow \psi)$
- $(M, s) \models \forall x\psi \iff (M, s) \models \psi[x/c]$ *for all* $c \in Ds$
- $(M, s) \models \forall x\psi \iff (M, s) \models \psi[x/c]$ *for some* $c \in Ds$
- $(M, s) \models \psi \wedge \varphi \iff (M, s) \models \psi$ *and* $(M, s) \models \varphi$
- $(M, s) \models \psi \vee \varphi \iff (M, s) \models \psi$ *or* $(M, s) \models \varphi$
- $(M, s) \models \Diamond\psi \iff (M, s') \models \psi$ *for some* $k \geq i$, *where* $s = (t, i)$ *and* $s' = (t, k)$
- $(M, s) \models \Box\psi \iff (M, s') \models \psi$ *for all* $k \geq i$, *where* $s = (t, i)$ *and* $s' = (t, k)$
- $(M, s) \models K_i\psi \iff \forall s' = (t', n'), t' \in TLs$ *and* $n' \in \mathbb{N} : (s, s') \in R_i \rightarrow (M, s') \models \psi$
- $(M, s) \models \left(\varphi \xrightarrow{refine} \psi\right) \iff \left((M, s) \models \varphi \leftrightarrow (M, s) \models \psi\right)$

The truth condition for $\perp$ are defined from those above. In particular, $\varphi \xrightarrow{refine} \psi$ can be expressed in term of $\varphi \leftrightarrow \psi$. For sake of clarity, $\varphi \xrightarrow{refine} \psi$ is used to distinguish between the axioms of our model and the refinement rules to be used to refine security policies.

For any formula Ψ of $\mathcal{L}$, if there is a model M and a timeline t such that $s = (t, 0)$ and $(M, s) \models \Psi$, then Ψ is said to be satisfiable. If for any formula Ψ of $\mathcal{L}$ and for any model M, there exists a timeline s such that $s = (t, 0)$ and $(M, s) \models \Psi$, then Ψ is said to be valid. Note that when using a temporal logic, satisfiability and validity are evaluated at the beginning of time (initial state of the target system) [Emerson 1995].

5.2.3 Axiomatizations

In this section, we provide a sound axiomatization of our language $\mathcal{L}$. First, we introduce a system $\mathcal{S}_\mathcal{L}$ that extends, to the first-order case, the multi-modal epistemic logic S5 combined with the linear temporal logic LTL.

Definition 19. *The system $\mathcal{S}_{\mathcal{L}}$ contains the following schemes of axioms and rules, where ϕ and ψ are formulas in $\mathcal{L}$, $\Rightarrow$ is the inference relation, and $\divideontimes$ is a placeholder for any primitive modality in $\mathcal{L}$ ($\Diamond$ and $\Box$, and K_i):*

- *First order logic*
  - ***Taut**: classical propositional tautologies;*
  - ***MP**: $\phi \rightarrow \psi, \phi \Rightarrow \psi$;*
  - ***Ex**: $\forall x\phi \rightarrow \phi[x/t]$*
  - ***Gen**: $\phi \rightarrow \psi[x/t] \Rightarrow \phi \rightarrow \forall x\psi$, where x is not free in ϕ;*
- *General Rules and Axioms*
  - ***Dist**: $\divideontimes(\phi \rightarrow \psi) \rightarrow (\divideontimes\phi \rightarrow \divideontimes\psi)$*
  - ***4**: $\divideontimes\psi \rightarrow \divideontimes\divideontimes\psi$*
  - ***Nec**: $\phi \Rightarrow \divideontimes\phi$*
- *Temporal Logic*
  - $\Box\neg\phi \leftrightarrow \neg\Diamond\phi$
- *Epistemic logic*
  - ***T**: $K_i\phi \rightarrow \phi$*
  - ***5**: $\neg K_i\phi \rightarrow K_i\neg K_i\phi$*

The operators $\Diamond$ and $\Box$ are axiomatized as linear-time modalities [Fagin et al. 1995], while K_i operator is an S5 modality. To this, we added the classical postulates *Ex* and *Gen* for quantification.

In our formal model, the standard definitions of *proof* and *theorem* are considered. $\vdash \phi$ means that $\phi \in \mathcal{L}$ is a theorem in $\mathcal{S}_{\mathcal{L}}$. A formula $\phi \in \mathcal{L}$ is *derivable* in $\mathcal{S}_{\mathcal{L}}$ from a set of formulas Σ, or $\Sigma \vdash \phi$, if and only if $\vdash \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_n \rightarrow \phi$ for some $\psi_1, \psi_2, \cdots, \psi_n \in \Sigma$.

It can be easily checked that the axioms and rules of $\mathcal{S}_{\mathcal{L}}$ are valid on every model of $\mathcal{L}$ and that the inference rules presented in Definition 19 preserve validity. Subsequently, we have the following soundness result.

Theorem 5. *The system $\mathcal{S}_{\mathcal{L}}$ is sound with respect to $\mathcal{L}$.*

Proof. Proof follows from soundness of S5 and soundness of the axiom system for first order temporal logic of time isomorphic to the set of natural numbers proved in [Szalas 1987]. □

For the rest of Chapters 5 and 6, we use $\mathcal{D}$ and $\mathcal{V}$ to denote respectively the set of objects in the data to be outsourced and the set of values assumed by those objects.

For the sake of simplicity, we will use the two places predicate $knows(i, o)$ to denote that the agent i knows the value of the data object o. The relation between the $knows$ predicate and the epistemic predicate K is described using the following axiom:

$$\forall o \in \mathcal{D}, \forall e : \ \Big(knows(e, o) \leftrightarrow \exists v \in \mathcal{V} : K_e value(o, v)\Big) \tag{5.1}$$

where $value(o, v)$ is used to denote that the value assumed by the object o is v.

5.3 Data Model Specification

Information is traditionally stored using different kind of data structure such as, relational databases, file systems, graphs, XML files, etc. One of our main objectives is to give the ability to the data owner to specify as finely as possible the security policy to be enforced over the data to be outsourced. To meet this goal, and relying on the fact that each data structure can be represented as a set of abstraction levels, we defined a general model that can specify the different abstraction levels and their corresponding involved objects for all existing data structures.

Tree-based Modelization of Data Structures

The intuition behind this idea is that the most of existing data structures can be expressed as a set of abstraction levels. In fact, through a simple analysis of the data structures traditionally used to store data, we realize that we can represent each data structure as a tree. For each data structure, each layer of the corresponding tree will represent an abstraction level involving the corresponding set of objects that belongs to it. For example, in a relational database, we have mainly four abstraction levels: (1) the database-level which represents all the information stored in the database, (2) table-level representing the set of relational tables in the database and the associations between them (the associations between relational tables can be considered as sensitive [Bkakria et al. 2013b]), (3) the attribute-level which contains the set of attributes of each relational table and the associations between them, and (4) the value-level containing the set of values stored in the database.

The tree-based modelization of data structures offers our approach two main advantages. First, it allows data owner to specify as finely as possible his\her security requirements over the data structure representing the data to be outsourced. The second advantage is that the relations between different abstraction layers in the tree can be used to propagate the policy to be enforced. Tree structures are composed of a set of nodes and a set of edges (Hierarchical relation between nodes). To be able to use the tree-based modelization in our approach, we define a two places predicate $subelement\_of(o, o')$ allowing to formally specify the branches of the tree. $subelement\_of(o, o')$ means that the object o' is a sub-object of o. Note that the predicate $subelement\_of$ is transitive, since obviously, if o_1 is a sub-object of o and that o_2 is a sub-object of o_1, then o_2 is a sub-object of o.

Example 10. *Assume that we have a relational database D of a medical insurance company which contains two relational tables Patient and Doctor represented respectively in Table 5.1 and Table 5.2. The tree-based modelization of the database D are represented in Figure 5.1.*

Table 5.1 – Patient relation

Name_pat	Illness	Id_doc
A. Barrett	Illness_1	doc_1
C. Beat	Illness_2	doc_2

Table 5.2 – Doctor relation

Id_doctor	Name_doc
doc_1	C. Amalia
doc_2	D. Annli

In the formal specification of the data structure, each node in the corresponding tree will be represented by an object (e.g., $object(D)$, $object(Patient)$, etc), and each edge linking two nodes will be formalized using the predicate $subelement\_of$ (e.g., $subelement\_of(Patient, Name\_pat)$, etc).

The transitivity of the the predicate $subelement\_of$ will allow us to define the rules (Definitions 20 and 21) make it possible to propagate the properties (e.g., sensitivity, knowledge, etc) of each object within the data to be oustourced to its sub-objects.

Definition 20. ***(refinement transitivity).*** *Given a predicate P in $\mathcal{L}$ describing a property (e.g., sensitivity, knowledge, etc) of a data object o, P is a refinement transitive predicate if and only if the following condition holds:*

$$\forall x_1, \cdots, x_n, y_1, \cdots, y_m, \forall o, o' \in \mathcal{D} : \Big(P(x_1, \cdots, x_n, o, y_1, \cdots, y_m) \wedge subelement\_of(o, o')\Big) \rightarrow P(x_1, \cdots, x_n, o', y_1, \cdots, y_m)$$

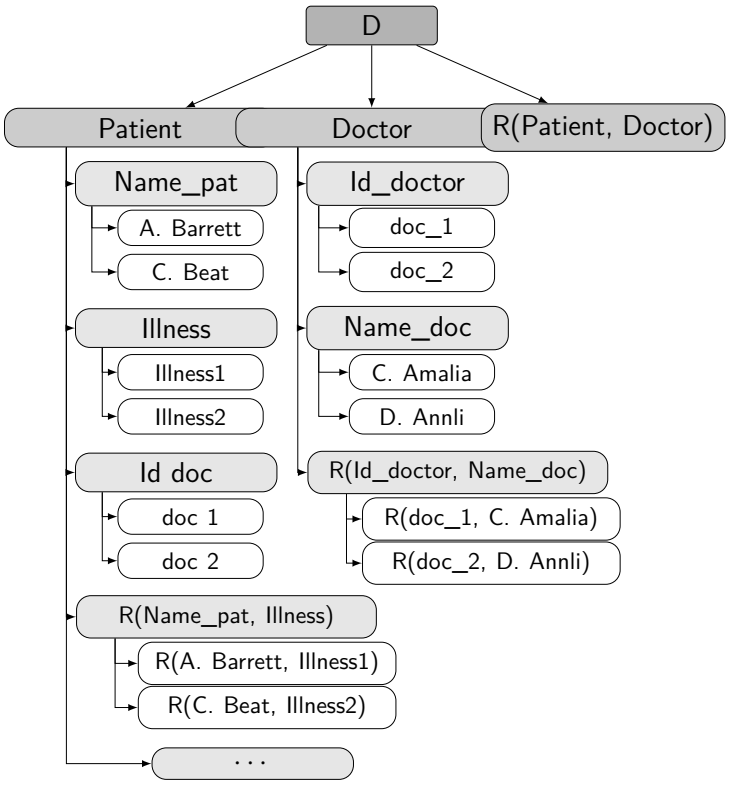


Figure 5.1 – A tree-based representation of the relational database D

Definition 21. ***(abstraction transitivity).*** *Given a predicate P in $\mathcal{L}$ describing a property (e.g., sensitivity, knowledge, etc) of a data object o, P is an abstraction transitive predicate if and only if the following condition holds:*

$$\forall x_1, \cdots, x_n, y_1, \cdots, y_m, \forall o, o' \in \mathcal{D} : \Big(P(x_1, \cdots, x_n, o', y_1, \cdots, y_m) \wedge subelement\_of(o, o')\Big) \rightarrow P(x_1, \cdots, x_n, o, y_1, \cdots, y_m)$$

Actually, as far as we will go further in depth and details of our formal model, we will see how much The *refinement transitivity* and *abstraction transitivity* relations are useful. In fact, they will allow us in a first time to propagate the properties of the data objects to more detailed or concrete levels' data objects, and in a second time to refine the goals to satisfy and the policy to apply to more detailed or concrete levels. The purpose behind refining goals and policies is that it is likely to happen that the policy to be enforced and the goals to be satisfied are specified in different levels of abstraction. Therefore, an efficient reasoning method over our formal model requires the ability to refine goals and policies to the same level of abstraction.

Definition 22. ***(Refinement Rule).*** *Given two well formed formulas φ and ψ in $\mathcal{L}$. A refinement rule is a formula having the following form:*

$$\varphi \xrightarrow{refine} \psi$$

Semantically speaking, according to Definition 18, a refinement rule $\varphi \xrightarrow{refine} \psi$ stating that ψ is a concrete level representation of φ means that if ψ (resp. φ) is satisfied in a state w of the target system ($(w, \mathcal{W}) \models \psi$, resp. $(w, \mathcal{W}) \models \varphi$) then φ (resp. ψ) is also satisfied in w ($(w, \mathcal{W}) \models \varphi$, resp. $(w, \mathcal{W}) \models \psi$).

5.4 Goal Specification

From a data owner point of view, two main objectives can be targeted when using a Cloud storage model: data outsourcing and data sharing.

- Data outsourcing: The data owner aims to outsource its data $\mathcal{D}$ to a Cloud storage provider (csp) server s. This objective can be specified as following:

$$\forall o \in \mathcal{D}, \exists s : \ \Diamond \ (csp(s) \wedge outsource(o, s)) \tag{5.2}$$

  The previous formula specifies that, eventually, any object o in $\mathcal{D}$ should be outsourced to a *csp* server s. Note that the two places predicate *outsource* is a *refinement transitive* predicate, Since if a data object o is outsourced, obviously all sub-objects of o are also outsourced, which allows data outsourcing goals to be refined to a more concrete level using the following rule:

$$\begin{array}{l}(\forall o \in \mathcal{D} : \ \Diamond \ outsource(o, csp)) \xrightarrow{refine} \\ (\forall o' \in \mathcal{D} : subelement\_of(o, o') \rightarrow \Diamond \ outsource(o', csp))\end{array} \tag{5.3}$$

  The relation between the knowledge of the information stored by a data object and the fact that it is outsourced to a *csp* is described through the following formula. $val(o, v)$ is satisfied if the value of the data object o is v.

$$\forall o \in \mathcal{D}, \forall s : (csp(s) \wedge outsource(o, s) \rightarrow knows(s, o)) \tag{5.4}$$

- Data sharing : The data owner wants to share all or a part of the outsourced data $\mathcal{D}$ with a set of entities E (users, servers, etc.). This kind of goal is specified using the following formula:

$$\forall e \in E, \forall \mathcal{D}' \subseteq \mathcal{D}, \forall o \in \mathcal{D}' : \ \Diamond \ knows(e, o). \tag{5.5}$$

Formula 5.5 specifies that in a future state of the target system, any entity in E should know the value of any object in $\mathcal{D}'$. The knowledge predicate *knows* is a *refinement transitive* predicate as if we consider that an entity e knows the information stored in an object o, therefore it knows the information stored by any sub-object o' of o ($subelement\_of(o, o')$). The propagation of the knowledge predicate *knows* allows us to refine a data sharing goal to a more concrete level using the following formula:

$$\forall e \in E, \forall \mathcal{D}' \subseteq \mathcal{D}, \forall o \in \mathcal{D}' : \Big((\Diamond\ knows(e, o)) \xrightarrow{refine} (\forall o' \in \mathcal{D} : subelement\_of(o, o') \rightarrow \Diamond\ knows(e, o')) \Big) \tag{5.6}$$

5.5 Policy Specification

The policy to be enforced over the target data model $\mathcal{D}$ is specified through a set of security and utility constraints.

5.5.1 Security Constraints

Using security constraints, the data owner specifies the security requirements that should be enforced over the data model to be outsourced. We define five types of security constraints.

Confidentiality constraint: It requires the protection of the confidentiality of an object in the target data model.

$$\Box\ [\forall o \in \mathcal{D}, \forall e :\ sensitive(o) \wedge untrusted(e) \rightarrow \neg knows(e, o)]. \tag{5.7}$$

Formula 5.7 specifies that in any state of the target system, an untrusted entity e should not know the information stored by any sensitive object o. The one place predicate *sensitive* is a *refinement transitive* predicate as we consider that if the data owner specifies that the information stored by an object o are sensitive, therefore the information stored by any sub-object o' of o will be also sensitive. This property allows us to refine a confidentiality constraint as described in the following formula:

$$\forall o \in \mathcal{D}, \forall e : \Big(\Box\ (\ sensitive(o) \wedge untrusted(e) \rightarrow \neg knows(e, o)) \xrightarrow{refine} \Box\ (\forall o' \in \mathcal{D} : subelement\_of(o, o') \wedge sensitive(o') \wedge untrusted(e) \rightarrow \neg knows(e, o')) \Big). \tag{5.8}$$

Privacy constraint: The data owner can use privacy constraints to require the prevention of identity disclosure.

$$\Box\ [\forall o \in \mathcal{D}, \forall e :\ identifier(o) \wedge untrusted(e) \rightarrow \neg knows(e, o)]. \tag{5.9}$$

Formula 5.9 specifies that an object o that can be exploited to identify an identity should not be known by any untrusted entity e in any state of the target system. In our formal model, we consider that the one place predicate $identifier$ is not a *refinement transitive* predicate, since we consider that only all the information stored by an identifier object o allow to identify a person (or entity) to which they belong. In other words, the disclosure of only some sub-objects of an identifier object might not lead to disclose the identity of the person to which the identifier object belongs. As a consequence, a privacy constraint cannot be refined.

Traceability constraint (Traitor detection): In the applications where databases contents are publicly available over a network, the contents owner would like to discourage unauthorized duplication and distribution of his valuable contents. To meet this goal, the owner wants to give to a set of entities E' the ability to check whether or not his valuable contents have been released to unauthorized users.

$$\begin{aligned}\Box\ [\forall o \in \mathcal{D}, \forall e :\ & sensitive(o) \wedge untrusted(e) \wedge knows(e, o) \rightarrow \\ & \bigwedge_{e_1 \in E'} K_{e_1}(\exists E_r.\ \bigwedge_{e_r \in E_r} (trusted(e_r) \wedge responsible(e_r, o)))].\end{aligned} \tag{5.10}$$

Formula 5.10 means that in any state of the system, if an untrusted entity knows a sensitive object o, the set of entities E' should be able to know the set of entities E_r responsible of the disclosure of the sensitive object o. The two places predicate $responsible(e_r, o)$ means that the entity e_r is responsible of the disclosure of the sensitive object o to the untrusted entity e. It is a *refinement transitive* predicate as we consider that if an entity e_r is responsible of the disclosure of the sensitive object o, then it is responsible of the disclosure of any sub-object of o. As a result, a traceability constraint can be refined to a more concrete level using the following formula:

$$\begin{aligned}\forall o, \forall e :\ \Big(\Box\ [\ & object(o) \wedge sensitive(o) \wedge untrusted(e) \wedge knows(e, o) \rightarrow \\ & \bigwedge_{e_1 \in E'} K_{e_1}(\exists E_r.\ \bigwedge_{e_r \in E_r} (trusted(e_r) \wedge responsible(e_r, o)))] \\ & \xrightarrow{refine} \\ \Box\ [\forall o' \in \mathcal{D} :\ & subelement\_of(o, o') \wedge sensitive(o') \wedge untrusted(e) \wedge \\ konws(e, o') \rightarrow & \bigwedge_{e_1 \in E'} K_{e_1}(\exists E_r.\ \bigwedge_{e_r \in E_r} (trusted(e_r) \wedge responsible(e_r, o')))]\Big).\end{aligned} \tag{5.11}$$

Ownership constraint: The data owner wants to give a set of entities E the ability to verify the ownership of an object in the target data model.

$$\begin{aligned}\Box\,[\forall o_1.o_2, e \in \mathcal{D} : copy\_of(o_1, o_2) \bigwedge_{e_r \in E} K_{e_r} owner(e, o_1) \\ \rightarrow \bigwedge_{e_r \in E} K_{e_r} owner(e, o_2)].\end{aligned} \tag{5.12}$$

Formula 5.12 specifies that in any state of the target system, if there are two objects o_1 and o_2 such that o_2 is a copy of o_1 and a set of entities E which know that the owner of o_1 is e, therefore E should be able to know that o_2 belongs to e. The two places predicate *owner* is also a *refinement transitive* predicate as if an entity e is the owner of an object o, then it is the owner of any sub-object of o. Formula 5.13 allows to refine an ownership constraint to a more concrete level.

$$\begin{aligned}\forall o_1, o_2 \in \mathcal{D} : \Big(\Box\,\Big[copy\_of(o_1, o_2) \bigwedge_{e_r \in E} K_{e_r} owner(e, o_1) \rightarrow \\ \bigwedge_{e_r \in E} K_{e_r} owner(e, o_2)\Big] \xrightarrow{refine} \Box\,\Big[\forall o_1', o_2' \in \mathcal{D} : subelement\_of(o_1, o_1') \\ \wedge\ subelement\_of(o_2, o_2')\ \wedge\ copy\_of(o_1', o_2') \bigwedge_{e_r \in E} K_{e_r} owner(e, o_1') \\ \rightarrow \bigwedge_{e_r \in E} K_{e_r} owner(e, o_2')\Big]\Big)\end{aligned} \tag{5.13}$$

Integrity assessment constraint: This kind of constraint allows the data owner requiring the insurance of the accuracy and consistency of an object o over its entire life-cycle. This means that data cannot be modified in an undetected manner. In the target system, we should be able to check if o has been modified or not. A data owner may want to give a set of entities E' the ability to check the integrity of an object o.

$$\Box\,\Big[\ object(o) \rightarrow \bigwedge_{e_1 \in E'} K_{e_1}(is\_modified(o) \vee is\_unmodified(o))\Big]. \tag{5.14}$$

In the first hand, the one place predicate $is\_modified$ is an *abstraction transitive* predicate, as if we consider that the information stored by an object o are modified, then the information stored by any over-object o' of o ($subelement\_of(o', o)$) will be also modified. In the other hand, the one place predicate $is\_unmodified$ is a *refinement transitive* predicate, since if we consider that the information stored by an object o are not modified, then we are sure that the information stored by any sub-object o' of o are

not modified. Formula 5.15 can be used to refine an integrity assessment constraint:

$$\begin{aligned} &\square \Big[\ object(o) \rightarrow \bigwedge_{e_1 \in E'} K_{e_1}(is\_modified(o) \vee is\_unmodified(o))\Big] \\ &\qquad\qquad \xrightarrow{refine} \\ &\square \Big[\ \forall o' \in \mathcal{D} : subelement\_of(o, o') \rightarrow \bigwedge_{e_1 \in E'} K_{e_1}(is\_modified(o') \vee \\ &\qquad\qquad\qquad\qquad is\_unmodified(o'))\Big] \end{aligned} \tag{5.15}$$

5.5.2 Utility Constraint

Generally, outsourced data protection is offered at the expense of data utility. In our model, we define a set of utility constraints giving the ability to a data owner to require that particular utility properties on the target data model must be respected. The violation of these properties makes the data useless. As we are working in the Cloud data model, utility requirements are properties allowing the data owner to efficiently use the outsourced data. These utility requirements can be classified into four classes.

Computational requirements. With this kind of requirements, a data owner wants to have the ability to perform computation efficiently over outsourced data. For example, in the case of relational outsourced databases, computational requirements means the ability to execute queries with SUM, AVG, etc.

Keyword search requirements. Using keyword search requirements, a data owner wants to have the ability to perform keyword based search over the outsourced data. For instance, in the case of an outsourced file-based data (e.g., Dropbox), Keyword search requirements means the ability to pickup files containing certain words.

Equality check requirements. With this kind of requirements, a data owner wants to be able to perform equality checks. For example, in the case of relational databases, equality check requirements means that the data owner wants to be able to perform data selections with equality predicates, equality joins, etc.

Order check requirements. A data owner can use this kind of requirement in order to perform order check, which means that he or she wants to have the ability to perform order searches over the data to be outsourced.

To be able to express these different kinds of utility requirements, we define the one place predicate *utility_requirement*(). Then, an utility constraint defined over a data object o can be expressed by the axiom 5.16, which is to be read: "the ability to perform the utility requirement req over the object o".

$$utility\_requirement(req) \ \wedge \ provides(req, o) \tag{5.16}$$

The one place predicate *provides* is a *refinement transitive* predicate, since we consider that if an utility requirement is provided for an object o, then it is also provided for any sub-object o' of o.

5.6 Mechanisms Specification

One of our main objective when designing our formal model is the ability to integrate the largest number of security mechanisms that has been defined to protect outsourced data. In fact, when studying those security mechanisms, we realize that they can be specified using two groups of formulas: preconditions formulas and effects formulas.

Preconditions. For each mechanism, preconditions are represented by a set of formulas which are necessary conditions required for the application of the security mechanism. We define the two-places predicated *is_applicable*. The formula $is\_applicable(m, o)$ is to be read "the mechanism m can be applied over the object o". Preconditions of a security mechanism m are specified using a formula of the following form:

$$\Box \ (is\_applicable(m, o) \ \rightarrow \ \Delta_m) \tag{5.17}$$

Where Δ_m represents necessary conditions for the applicability of the mechanism m. A formula of the form 5.17 is to be read "At any state of the target system, m can be applied if the preconditions Δ_m hold".

Effects. They are modifications resulting from the application of a mechanism m that transits the system from a state s to a state s'. We use the two-places predicate $apply(m, o)$ to specify that the mechanism m is applied over the object o. For a mechanism m, effects are represented by a set of formulas Σ_m such that:

$$\Phi(s, \ apply(m, o)) = s' \ \rightarrow \ (s' \ \models \ \Sigma_m) \tag{5.18}$$

Axiom 5.18 states that if the application of the mechanism m over the object o transits the system from a state w_i to a state w_j, therefore the set of effects Σ_m of the application of the mechanism m is satisfied in the state w_j.

Many security mechanisms have been proposed to ensure outsourced data protection. In the present work, we are going to focus mainly on four classes of security mechanisms. Each class ensures specific security and utility properties. In each of these classes, we studied some security mechanisms to figure out (1) the data structure(s), the data type(s), and the level of granularity over which they can be applied, (2) the set of provided security and utility properties. A quick overview of the characteristics of the studied security mechanisms is given in Table 5.3.

5.6.1 Encryption Based Security Mechanisms

In our model, we consider that encryption-based security mechanisms can be applied over a data object o if and only if the following three conditions hold: (1) the information stored by o are considered sensitive (e.g., confidential or private information), (2) the information stored by o are typed t, and (3) the information stored by o should be structured as ds. This can be specified as follows:

$$\begin{aligned}\Box\Big[&enc\_mechanism(m) \wedge is\_applicable(m,o) \rightarrow object(o) \\ &\wedge sensitive(o) \wedge data\_type(o,t) \wedge data\_structure(o,ds)\Big]\end{aligned} \tag{5.19}$$

For instance, for the searchable encryption based security mechanism [Moataz and Shikfa 2013], it is applicable over an object o if its stored information are considered sensitive and having a *textual* data type ($data\_type(o, textual)$), and this, no matter how the information stored in o are structured.

The effects of the application of encryption-based mechanisms are specified using the following formula:

$$\begin{aligned}\forall m, o, k : enc\_mechanism(m) \wedge apply(m,o) \wedge enc\_key(k) \rightarrow \\ \exists o_e : encrypted(o,k,o_e) \bigwedge_{p \in P_m} provides(p, o_e)\end{aligned} \tag{5.20}$$

where P_m represents the set of utility properties provided by the security mechanism m. For instance, the set of utility properties provided by the order preserving encryption [Boldyreva et al. 2009] is $P_{ope} = \{ES, OS\}$ (according to Table 5.3). The three places predicate *encrypted* is a *refinement transitive* predicate as we consider that if the data stored by an object o are encrypted using an encryption key k, therefore the information stored by any sub-object o' of o will be also encrypted using k.

Approaches	Mechanisms	Data Structures	Data Types	Granularity	Security Properties					Utility Properties				
					Co	Pr	In	Tr	Ow	ES	OS	KS	Cp	SO
Encryption	[Boldyreva et al. 2009]	Any	Numeric	Any	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗
	[Pohlig and Hellman 1978]		Any		✓	✓	✗	✗	✗	✓	✗	✗	✗	✗
	[Paillier 1999]		Numeric		✓	✓	✗	✗	✗	✗	✗	✗	✓	✗
	AES-CBC		Any		✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
	[Moataz and Shikfa 2013]		Textual		✓	✓	✗	✗	✗	✗	✗	✓	✗	✗
	[Tseng et al. 2013]	Databases	Any	Attribute level	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓
Anonymization	[Sweeney 2002]	Relational Database	Categorical data	Table level	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
	[Machanavajjhala et al. 2006]													
	[Li et al. 2007]													
Watermarking	[Guo et al. 2006]	Any	Numeric	Bits-level	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓
	[Zhang et al. 2006]	Database			✗	✗	✓	✗	✓	✓	✓	✓	✓	✓
	[Pournaghshband 2008]	Database table	Any	Attribute-level	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
	[Bhattacharya and Cortesi 2009]	Tuple-based structure		Tuple-level	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓
	[Charpentier et al. 2011]	Video	Binary	Bits-level	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓
Fragmentation	[Ciriani et al. 2007]	One-relation Database	Any	Attribute-level	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
	[Bkakria et al. 2013b]	Multi-relation Database												

Co: Confidentiality preserving of outsourced data.
Pr: Privacy preserving of outsourced sensitive data.
In: Integrity preserving of outsourced data.
Tr: Traceability insurance of outsourced data.
Ow: Ownership insurance of outsourced data.
ES: The ability to perform equality search over the outsourced data (*).
OS: The ability to perform order search over the outsourced data (*).
KS: The ability to perform keyword-search over the outsourced data (*).
Cp: The ability to perform computation over the outsourced data (*).
SO: The ability to perform statistical operations over the outsourced data (*).
*: Without the need to download all the outsourced data.

Table 5.3 – A quick overview of the characteristics of the studied security mechanisms

The relation between the knowledge of the information stored by data objects and the property *encrypted* produced by the application of an encryption based security mechanism is depicted through the following hypothesis.

Hypothesis 1. *Given a data object o and an encryption key k. If we suppose that the information stored by o are encrypted using k and stored into the data object o_e then, the following formula is an axiom in our model:*

$$\begin{aligned} \forall o, \forall o_e, \forall k, \forall e : enc\_key(k) \wedge encrypted(o, k, o_e) \rightarrow \\ \Big(knows(e, o) \leftrightarrow knows(e, o_e) \wedge knows(e, k)\Big) \end{aligned} \tag{5.21}$$

The previous Hypothesis states that an entity e knows the unencrypted information stored in a data object o if and only if it knows the encrypted information o_e of o and the encryption key k.

5.6.2 Anonymization Based Security Mechanisms

Anonymization based security mechanisms are traditionally used to prevent identity disclosure when publishing data sets. As a consequence, in our formal model, we formally specify (Formula 5.22) that an anonymization based security mechanism can be applied over a data object o if and only if: (1) the information stored by o are considered to be exploited to identify an entity's identity, (2) the information stored by o are typed t, (3) the information stored by o should be structured as an existing data structure ds, and (4) the information stored by o are not encrypted [1].

$$\begin{aligned} \nexists k, o' : \Box\Big[anonym\_mechanism(m) \wedge is\_applicable(m, o) \rightarrow object(o) \\ \wedge identifier(o) \wedge encrypted(o', k, o) \wedge data\_type(o, t) \wedge data\_structure(o, ds)\Big] \end{aligned} \tag{5.22}$$

For illustration purpose, let us take the case of *k-anonymity* ([Sweeney 2002]), it can be applied over a data object o if the information stored by o are *categorical data* ($data\_type(o, categorical)$) and structured as a relational table ($data\_structure(o, relational\_table)$).

[1] Anonymization-based mechanisms are based on data generalization consisting on replacing each value in the object to be anonymized with a broader category, however this is will not be feasible if those values are encrypted.

The effects of the application of anonymization-based security mechanisms are specified by the following formula:

$$\forall m, o : anonym\_mechanism(m) \wedge apply(m, o) \rightarrow \exists o_a : anonymized(o, o_a) \bigwedge_{p \in P_m} provides(p, o_a) \tag{5.23}$$

where P_m represents the set of utility properties provided by m. We consider that the two places predicate *anonymized* is a *refinement transitive* predicate since if the data stored by an object o are anonymized, then any identifier sub-object of o is also anonymized. To illustrate, let us consider that the object o is represented by a relational database. if we suppose that o is anonymized, then obviously, any relational table in o is anonymized.

By considering that a data object o is identifier, we are sure that if the information stored by o are known by an entity e, then e knows the set of identities id_o related to o. However if the information stored in o are anonymized and stored in o_a, then any entity, which does not know the information stored in o, cannot know the set of identities id_o related to o even if he or she has knowledge of the anonymized information stored in o_a. This can be formalized in our model through the following hypothesis.

Hypothesis 2. *Given a data object o, if we suppose that the information stored by o are anonymized and stored in o_a, then the following formula is an axiom in our model:*

$$\forall o, o_a, e, id : \neg knows(e, o) \wedge knows(e, o_a) \wedge anonymized(o, o_a) \rightarrow \neg(knows(e, id) \wedge id\_related(o, id)) \tag{5.24}$$

5.6.3 Watermarking Based Security Mechanisms

Digital watermarking of outsource data emerged as a candidate solution for providing ownership protection, integrity maintaining, and traitor tracing. Many watermarking techniques have been proposed in the literature to address these purposes.

In our formal model, the conditions under which a watermarking-based security mechanism can be used are specified using Formula 5.25.

$$\Box\Big[watermark\_mechanism(m) \wedge is\_applicable(m, o) \rightarrow object(o) \wedge data\_type(o, t) \wedge data\_structure(o, ds)\Big] \tag{5.25}$$

The data type t and the data structure ds are the properties that the object o should have to allow the application of the watermarking-based security mechanism m. For

example, the watermarking-based security mechanism presented in [Zhang et al. 2006] can be applied over a data object o if o is typed numeric ($t = numeric$) and the information stored in o are represented as a database ($ds = database$).

The effects of the application of watermarking-based security mechanisms are specified using the following formula:

$$\begin{aligned}&\forall m, o, k : watermark\_mechanism(m) \wedge apply(m, o) \wedge watermark\_key(k) \\ &\rightarrow \exists w, o_w : watermark(w) \wedge watermarked(o, w, k, o_w) \bigwedge_{p \in P_m} provides(p, o_w)\end{aligned} \tag{5.26}$$

Watermarking-based security mechanisms can be classified along two main dimensions regarding the Verifiability/Detectability used method. Watermark extraction is *blind* if and only if any entity needs only some private parameters (e.g. secret key) to be able to extract the watermark. Watermark extraction is *non blind* if and only if the knowledge of the original unwatermarked database is required. These two properties are specified respectively by Formula 5.27 and 5.28 in Hypothesis3. We use the one place predicate $blind(m)$ to specify that the watermarking-based security mechanism m supports the blind Verifiability/Detectability method.

Hypothesis 3. *Given a watermarking-based security mechanism m, a data object o, and a watermark key k. The two following formulas are axioms in our model:*

$$\begin{aligned}&\forall m, o, k, w, o_w, e : watermark\_mechanism(m) \wedge watermark\_key(k) \\ &\wedge blind(m) \wedge watermark(w) \wedge watermarked(o, w, k, o_w) \rightarrow \\ &\Big(knows(e, o_w) \wedge knows(e, k) \leftrightarrow knows(e, w) \wedge K_e\ watermarked(o, w, k, o_w)\Big)\end{aligned} \tag{5.27}$$

$$\begin{aligned}&\forall m, o, k, w, o_w, e : watermark\_mechanism(m) \wedge watermark\_key(k) \\ &\wedge \neg blind(m) \wedge watermark(w) \wedge watermarked(o, w, k, o_w) \rightarrow \\ &\Big(knows(e, o_w) \wedge knows(e, k) \wedge knows(e, o) \leftrightarrow knows(e, w) \wedge \\ &K_e\ watermarked(o, w, k, o_w)\Big)\end{aligned} \tag{5.28}$$

Actually, the watermarks embedded in data objects can be classified regarding the security properties they provide in three classes:

- Ownership watermark: Information allowing to prove the identity of the owner of the data object are embedded in the watermark. We use the one place predicate $ownership\_watermark(w)$ to specify that w is an ownership watermark and the two places predicate $contain(w, id)$ to specify that the information contained in w are related to the identity id.

- Integrity watermark: Information allowing to verify the integrity of a data object are embedded in the watermark. We use the one place predicate $integrity\_watermark(w)$ to denote that w is an integrity watermark.
- Traceability watermark: Information allowing to prove the identity of the person or entity to whom a data object has been distributed are embedded in the watermark. The one place predicate $traceability\_watermark(w)$ is used to specify that w is a traceability watermark. We use the previously defined predicate $contain(w, id)$ to denote that the information contained in w are related to the identity id.

Hypothesis 4. *The following formula are axioms in our formal model.*

$$\begin{aligned} \forall o, k, w, o_w, e, id : watermark\_key(k) \wedge ownership\_watermark(w) \wedge \\ K_e\ watermarked(o, w, k, o_w) \wedge knows(e, w) \wedge contain(w, id) \rightarrow \\ K_e\ owner(id, o) \end{aligned} \tag{5.29}$$

$$\begin{aligned} \forall o, k, w, o_w, e, id : watermark\_key(k) \wedge integrity\_watermark(w) \wedge \\ K_e\ watermarked(o, w, k, o_w) \wedge knows(e, w) \rightarrow K_e\ (is\_modified(o) \\ \vee\ is\_unmodified(o)) \end{aligned} \tag{5.30}$$

$$\begin{aligned} \forall o, k, w, o_w, e, id, e' : watermark\_key(k) \wedge traceability\_watermark(w) \wedge \\ K_e\ watermarked(o, w, k, o_w) \wedge knows(e, w) \wedge knows(e', o_w) \wedge \\ untrusted(e')\ \wedge contain(w, id) \rightarrow K_e\ responsible(id, o) \end{aligned} \tag{5.31}$$

Axiom 5.29 specifies how the ownership property is ensured using a watermarking-based security mechanism. Formally speaking, if an entity e knows that an ownership watermark w is embedded into a data object o, then e knows that the object o is owned by the person or the entity having the identity related to w. Axiom 5.30 specifies the integrity assessment property provided by the use of a watermarking-based security mechanism. Formally speaking, if an entity e knows that an integrity watermark w is embedded into a data object o, then e can know whether or not the object o has been modified. Finally, axiom 5.31 formally describes the traceability property ensured through the application of a watermarking-based security mechanism. Formally speaking, if an entity e knows that a traceability watermark is embedded in a data object o and that the watermarked object o_w is known by an untrusted entity, then e is able to figure out the identity of the person or entity responsible of the disclosure of o_w.

5.6.4 Fragmentation Based Security Mechanisms

In many data structures (e.g., relational database), associations between values of various data objects are most sensitive than the values themselves. The idea then is to use fragmentation to break the sensitive relationships between those values.

In our formal model, the conditions under which a fragmentation-based security mechanism can be used are specified using Formula 5.32.

$$\begin{aligned}\Box\Big[&fragmentation\_mechanism(m) \wedge is\_applicable(m,o) \rightarrow \exists o_1,o_2,o_3 : \\ &(\bigwedge_{o' \in \{o_1,o_2,o_3\}} subelement\_of(o,o')) \wedge association(o_1,o_2,o_3) \wedge sensitive(o_3) \\ &\wedge data\_structure(o,ds)\Big]\end{aligned} \tag{5.32}$$

The predicate $association(o_1, o_2, o_3)$ specifies that o_3 represents the association between o_1 and o_2. Formally speaking, Formula 5.32 states that a fragmentation-based security mechanism is applicable over a data object o if (1) there exists three data objects sub-objects of o, where one of them represents a sensitive association between the two others, and (2) the information stored in o are structured as ds. For instance, for the fragmentation-based security mechanism presented in [Bkakria et al. 2013b], ds will be a *multi-relational database*, and o_1, o_2, and o_3 will be either relational tables or attributes.

The effects of the application of fragmentation-based security mechanisms are specified using the following formula:

$$\begin{aligned}&\forall m,o : fragmentation\_mechanism(m) \wedge apply(m,o) \rightarrow \Big(\forall o_1,o_2,o_3 : \\ &(\bigwedge_{o' \in \{o_1,o_2,o_3\}} subelement\_of(o,o')) \wedge association(o_1,o_2,o_3) \wedge sensitive(o_3) \\ &\rightarrow \exists f_1,f_2 : \Big((f_1 \neq f_2) \wedge (o_1 \in f_1) \wedge (o_2 \in f_2) \wedge (o_3 \notin f_1) \wedge (o_3 \notin f_2)\Big) \wedge \\ &\Big(\forall o', \exists k : subelement\_of(o,o') \wedge sensitive(o') \rightarrow encrypted(o',k,o'_e)\Big) \\ &\Big(\forall o'' : subelement\_of(o,o'') \wedge \neg sensitive(o'') \rightarrow \bigwedge_{p \in P_m} provides(p,o'')\Big)\Big)\end{aligned} \tag{5.33}$$

Formula 5.33 formally states that if a fragmentation-based security mechanism m is applied over a data object o, then (1) each two sub-objects of o associated through a sensitive association will be stored in two different data fragments to break the sensitive association. (2) All sensitive sub-objects of o will be encrypted to allow only authorized entities (to whom the encryption key will distributed) to access them. (3) the utility properties provided by m are ensured only for all insensitive sub-objects of o since o's sensitive sub-objects will be encrypted.

5.7 Conclusion and Contributions

In this chapter, we define a formal model using a well-founded formal language based on liner temporal epistemic logic allowing us, in a first time, to formally specify the data structure storing the information to be outsourced relying on the use of a tree-based modelization of data structures, as a second time, to specify as finely as possible the policy to be applied over the data to be outsourced, and in a third time, formally specify a set of candidate security mechanisms that can be used to enforce the specified policy, by formally describing for each security mechanism, the conditions under which they can be applied, the effects representing the system's changes due to the application of the mechanism, and the provided security and utility properties.

In the next chapter, we present a reasoning method for our formal model allowing to formally identify the relevant combination of mechanisms to efficiently enforce the defined security policy.

CHAPTER 6

Formal Reasoning Method to enforce Security Policies over Outsourced Data

With outsourced data protection in mind, many security mechanisms that allow us to ensure particular security requirements (e.g., confidentiality, integrity, etc.) have been defined. In this context, several important questions need to be investigated: (1) how to choose the right security mechanisms that should be used to enforce security policies defined by the owners of the data to be outsourced, (2) over which parts of the outsourced data the chosen security mechanisms should be applied, and (3) how to verify then, whether or not, the chosen security mechanisms actually enforce the defined security policies.

In this chapter, we strive to design a reasoning method for our formal model presented in Chapter 5. That is, relying on the target system formalization (i.e., the data structure to be outsourced and the entities involved in the Cloud storage model), the security policy formalization, and the security mechanisms properties formalization, we strive to provide a reasoning method that formally identifies the relevant combination of mechanisms to efficiently enforce the defined security policy.

The reminder of this chapter is as follows. Section 6.1 discusses the related work. Section 6.2 defines a four steps reasoning method that permits to choose, from existing security mechanisms, the ones that can satisfy the policy defined over the target system. Section 6.3 describes a GraphPlan-based method that uses the set of security mechanisms given by our four steps reasoning method to produce a near-optimal security mechanisms execution plan that enforces the security and utility requirements while offering the best trade-off between security, utility and complexity. Section 6.4 reports

the implementation and evaluation of our approach. Finally, Section 6.5 concludes this Chapter.

6.1 Related Work

Our reasoning method aims to find a combination of security mechanisms that enforces a set of security and utility constraints while reaching a chosen goal (i.e., data outsourcing or publishing).

Few research efforts have investigated how to combine security mechanisms to enforce security policies over outsourced data. One of the firsts attempt is proposed in [Ciriani et al. 2007], it consists of combining data fragmentation together with encryption to protect outsourced data confidentiality and can only be applied over one-relation databases [1]. In Chapter 3, we improved the approach presented in [Ciriani et al. 2007] in such a way that it can deal with multi-relation databases. We also proposed a secure and effective technique for querying data distributed in several service providers and improve the security of our querying technique in order to protect data confidentiality under a collaborative Cloud storage service providers model. Popa et al. in [Popa et al. 2011] and we in Chapter 4 have proposed approaches based on adjustable encryption. That is, different encryption schemes are combined to get the best trade-off between data confidentiality and data utility for outsourced relational databases. Cancellaro et al [Cancellaro et al. 2011] and Boho et al. [Boho et al. 2013] have proposed interesting approaches combining watermarking and encryption to protect both the confidentiality and traceability of outsourced multimedia files. All previously cited approaches have three main limitations: First, they are defined in such a way that only two pre-selected security mechanisms can be combined together. Second, they cannot deal with all security properties that can be required by data owners as each approach can provide at most two security properties. Finally, they cannot deal with all data structures that can be used to store outsourced data.

To the best of our knowledge, a single formal model was proposed to combine security mechanisms to address how data is used after it is released. Pretschner et al. [Pretschner et al. 2008] present a formal model of usage control mechanisms combination that formalizes the access control problem domain at a realistic level of complexity. In the proposed model, mechanisms are specified as trace transformers that map attempted events into actual usage controlled events. The model allows the specification of a wide range of usage control mechanisms.

[1] Databases composed of only one table

The problem of security mechanisms planning to enforce security policies while attempting to satisfy a set of goals was not widely investigated. The first attempt was proposed by Irwin et al [Irwin et al. 2008]. They investigate how a planning system, that uses security policies to ensure that planned actions will be able to occur, could leak sensitive information. They formally define information leakage within the used context and present two planning techniques which can be used to mitigate or eliminate this information leakage and prove their security. Afterwards, Bartoletti et al. [Bartoletti et al. 2009] proposed a static approach to study the composition of services while respecting a given security requirements. To this end, they extend the λ-calculus [Rosser 1984] with primitives for selecting and invoking services that enforce the given security requirements.

Planning graph analysis idea has been used with security purpose in mind. Armando et al. [Armando and Compagna 2002] proposed a fully automatic model that translates security protocol specifications into propositional logic to permit to effectively find attacks to protocols. The proposed model is a combination of (1) a reduction of protocol insecurity problems to planning problems and (2) well-known SAT-reduction techniques which have been used for planning. Subsequently, in [Armando et al. 2003] authors improved the approach proposed in [Armando and Compagna 2002] by exploring the application of a sophisticated SAT-reduction technique, Graphplan-based encoding, which has been used with success in AI planning. Later on, Armando et al. [Armando et al. 2014] proposed a model checker based on SAT solver for security-critical systems. The model checker relies on a successful combination of encoding techniques originally developed for planning with techniques developed for the analysis of reactive systems. The proposed approach can be applied in a variety of application domains (e.g., security protocols verification and security-sensitive business processes). Unfortunately, the previously mentioned approaches are developed to support the verification of security-critical systems and cannot be used to deal with the problem we aim to tackle in this chapter.

6.2 Choosing the Right Security Mechanisms

The right mechanism or combination of mechanisms is the one that fits in the best way the sets of security and utility constraints. As we have seen in the section 5.6, each security mechanism offers different level of protection and different kind of utility properties. The main challenge then is to choose the best mechanisms that satisfy chosen goals while enforcing defined security polices. Actually, in the general case of data outsourcing, security issues come when data is outsourced to an untrusted storage

service provider. Before sending the data to a Cloud storage server, security constraints are normally satisfied, as we can consider that, since the data is not outsourced, there is no security issues to worry about. Based on this, we define several steps allowing to choose the combination mechanisms that can be used to enforce the requirements of outsourced data owners.

First Step: Satisfying the Chosen Goals

In this first step of our reasoning method, the purpose is to find the set of mechanisms that allow data owners to reach their goals.

Definition 23. ***(Goal Satisfier).*** *Given a set of formulas Σ_G representing a goal G to achieve, a set of formulas Σ representing the specification of a target system $\mathcal{S}$, and a mechanism m represented by (Δ_m, Σ_m). m satisfies G in $\mathcal{S}$ iff the following two conditions hold:*

1. $\Sigma \models \Delta_m$
2. $\Sigma \cup \Sigma_m \vdash \Sigma_G$

Formally speaking, condition (1) ensures that Σ is a model of Δ_m (which represents the preconditions of m). That is, in the current state of the system $\mathcal{S}$, the mechanism m is applicable. Condition (2) states that from the specification of the used system Σ and the set of formulas Σ_m representing the effects of the mechanism m, we should be able to formally deduce the set of formulas Σ_G representing the goal G.

Example 11. *Consider a data owner o who wants to outsource an object ob to a cloud server storage s. Let us suppose that the mechanisms toolbox that can be used by the data owner contains the http_post mechanism which is represented by the following two formulas:*

$$\Delta_{http\_post} = \Big\{\exists s : \Big(csp(s) \wedge connected(o, s, 80) \wedge stores(o, ob)\Big)\Big\} \tag{6.1}$$

$$\Sigma_{http\_post} = \Big\{\exists s : \Big(csp(s) \wedge outsourced(ob, s)\Big)\Big\} \tag{6.2}$$

Formula 6.1 specifies that the http_post mechanism is applicable over an object ob by the data owner o if it stores ob, and there exists a cloud storage server (csp) s with which the data owner is connected. Formula 6.2 states that if the http_post mechanism is applied over an object ob, then ob will be outsourced to a server s. In fact, since o is the data owner of ob, we can say that o stores ob. Then, if we suppose that o is

already connected through the HTTP protocol with a cloud server s, we can deduce the following:

$$\Sigma \models \exists s : csp(s) \wedge connected(o, s, 80) \wedge stores(o, ob) \tag{6.3}$$

Then, based on formulas 6.1 and 6.3 we deduce that:

$$\Sigma \models \Delta_{http\_post} \tag{6.4}$$

The goal G of the data owner consists in outsourcing the object ob to a csp. This goal is specified in our model using the formula 5.2. Then, based on formulas 5.2 and 6.2 we deduce that:

$$\Sigma \ \cup \ \Sigma_{http\_post} \vdash \ \Sigma_G \tag{6.5}$$

Finally, relying on formulas 6.4 and 6.5 as well as Definition 23 we deduce that the mechanism http_post can satisfy the desired goal G of the data owner.

Second Step: Violated Security Constraints

After getting the set of mechanisms $\mathcal{M}$ that can be applied to satisfy the chosen goal, we start looking for each mechanism $m \in \mathcal{M}$ the set of violated security and utility constraints.

Definition 24. *Given a set of formulas Σ representing the specification of a target system $\mathcal{S}$, a mechanism $m \in \mathcal{M}$ represented by (Δ_m, Σ_m), and a security constraint C formalized using the set of formulas Σ_C. The constraint C is violated while the chosen goal G is satisfied iff the following condition holds:*

$$\Sigma \ \cup \ \Sigma_m \ \cup \ \Sigma_C \ \vdash \ \ \bot \tag{6.6}$$

Example 12. *Consider that we have the same scenario proposed in Example 11, except in this example, the data owner considers that: (1) the object ob to be outsourced consists of sensitive information, and (2) all csps are untrusted entities. As a consequence, no matter which csp will be used to outsource ob, it should not know any information about ob. The previous security requirement is in fact an instance of a confidentiality constraint C which can be specified, according to Formula 5.7, using Σ_C^1 as following:*

$$\Sigma_C^1 = \{\Box \ [\forall s : \ sensitive(ob) \wedge untrusted(s) \rightarrow \neg knows(s, ob)]\}. \tag{6.7}$$

Based on (1) and (2) we deduce that:

$$\Sigma \ \models \ \Big(\forall s : csp(s) \ \wedge \ untrusted(s)\Big) \ \wedge \ sensitive(ob) \tag{6.8}$$

We have seen in Example 11 that the http_post mechanism can be used to satisfy the desired goal G of the data owner. Then, relying on Formulas 5.2 and 5.4, we can deduce the following:

$$\Diamond\Big(\exists s : csp(s) \wedge knows(s, ob)\Big) \tag{6.9}$$

Finally, based on Formulas 6.7, 6.8, and 6.9, we deduce that:

$$\Sigma \ \cup \ \Sigma_{http\_post} \ \cup \ \Sigma_C^1 \ \vdash \ \ \bot \tag{6.10}$$

Obviously, the toolbox to be used may contain several mechanisms that can satisfy the chosen goal of the data owner. In that case, our reasoning method should be able to choose the best one.

Definition 25. ***(Best Goal Satisfier).*** *Given the set of mechanisms $\mathcal{M} = \{m_1, \cdots, m_n\}$ that can be used to satisfy the defined goal G. Let $\mathcal{C}_i$ be the set of violated constraints while applying the mechanism m_i and $\mathcal{I}ns_i$ be the set of violated instances of the constraints $\mathcal{C}_i$ $(1 \leq i \leq n)$. The best goal satisfier m_{bgs} is defined as following:*

$$m_{bgs} = \left\{ \begin{array}{lll} m_j & if & \forall i \in \{1, \cdots, n\} : \ |\mathcal{C}_j| < |\mathcal{C}_i| \\ m_j & if & \exists p_1, \cdots, p_k \in \{1, \cdots, n\} : |\mathcal{C}_{p_1}| = |\mathcal{C}_{p_2}| = \cdots = |\mathcal{C}_{p_k}| \\ & & and \quad \forall i \in \{p_1, \cdots, p_k\} : |\mathcal{I}ns_j| \leq |\mathcal{I}ns_i| \end{array} \right\}$$

where $|\mathcal{C}|$ is used to denote the cardinality of $\mathcal{C}$.

Example 13. *Consider that we have the same scenario proposed in Example 12, except in this example, we consider that the mechanisms toolbox that can be used by the data owner contains both http_post and https_post mechanisms. The http_post mechanism is specified using the two formulas 6.1 and 6.2. The https_post mechanims is specified using the following two groups of formulas:*

$$\Delta_{https\_post} = \Big\{\exists s : \Big(csp(s) \wedge connected(o, s, 443) \wedge stores(o, ob)\Big)\Big\} \tag{6.11}$$

$$\begin{aligned} \Sigma_{https\_post} = \Big\{\exists s, k : \Big(csp(s) \wedge enc\_key(k) \wedge encrypted(ob, k, ob_e) \ \wedge \\ knows(s, k) \wedge \ knows(o, k) \wedge outsourced(ob_e, s)\Big)\Big\} \end{aligned} \tag{6.12}$$

We consider also that there is an untrusted adversary "adv" how is able to intercept the communication exchanged between the data owner and a csp. As a consequence, the data owner wants to prevent "adv" from getting the information stored in the object ob. The security requirements of the data owner can be specified as a confidentiality

constraint (Formula 5.7) which will be denoted in this example by C. Actually, we formally proved in Example 11 that the http_post allows to reach the desired goal of the data owner. The same approach can be used to formally prove that the https_post also allows to reach the same goal. In a first hand, in Example 12 we formally proved that, when the http_post is used to send the object ob to a csp, an instance (Formula 6.7) of the constraint C will be violated. In fact as we have supposed that 'adv' is able to intercept the communication exchanged between the data owner and a csp, we deduce the following:

$$\Diamond\ knows(adv, ob) \tag{6.13}$$

Then relying on Formula 6.13 and using the same approach as in Example 12, we deduce that a second instance (Formula 6.14) of C will be also violated when using the http_post mechanism.

$$\Sigma_C^2 = \{\Box\ [sensitive(ob) \wedge untrusted(adv) \rightarrow \neg knows(adv, ob)]\}. \tag{6.14}$$

In the other hand, when we consider that the https_post mechanism is to be used to outsource ob, we can rely on Formulas 5.2, 5.4, 6.12, and Hypothesis 1 to deduce 6.9. Then based on Formulas 6.7, 6.8, and 6.9, we deduce that the instance of C represented using Σ_C^1 is violated when using the https_post mechanism:

$$\Sigma\ \cup\ \Sigma_{https\_post}\ \cup\ \Sigma_C^1\ \vdash\ \ \bot \tag{6.15}$$

Since its considered that an external adversary cannot know the key used in an https communication. Then, it cannot know any information about ob just by intercepting the communication exchanged between the data owner and a csp.

As a conclusion, we deduce that, either using http_post or https_post mechanisms, the confidentiality constraint C will be violated. However we have seen also that the use of the http_post mechanism will violate two instances of C (both csp and adv will know the sensitive information stored in ob) whereas the use of the https_post mechanism will violate a single instance of C (only the csp will know the sensitive information stored in ob). Then according to Definition 25, the https_post mechanism is best goal satisfier in the used scenario.

Third Step: Satisfying the Violated Constraints.

Once we get the *best goal satisfier* m_{bgs} for a defined goal G and the corresponding set of violated security and utility constraints $\mathcal{C}$, the challenge then is to, for each violated security constraint, looking for the properties (e.g., encryption, anonymization, watermarking, computation, etc) that can satisfy that constraint.

Definition 26. *Given a set of properties $\mathcal{P} = \{P_1, \cdots, P_l\}$ specified respectively in our formal model using the sets of formulas $\Sigma_{P_1}, \cdots, \Sigma_{P_l}$, a set of formulas Σ representing the specification of the target system, a set of formulas $\Sigma_{m_{bgs}}$ representing the effects of m_{bgs}, and a violated constraint C represented in our model using the set of formula Σ_C. The set of properties $\mathcal{P}$ satisfies C iff the following condition holds:*

$$\bigwedge_{P \in \mathcal{P}} \Sigma_P \ \cup \ \Sigma \ \cup \ \Sigma_{m_{bgs}} \ \vdash \ \Sigma_C \tag{6.16}$$

Informally speaking, Formula 6.16 means that if the set of properties $\mathcal{P}$ is already provided, the application of the m_{bgs} will not lead to the violation of the security constraint C.

Example 14. *We consider that we will use the same scenario as in Example 13 in which we formally proved that the https_post mechanism is the best goal satisfier, despite the fact that it violates the instance Σ_C^1 (Formula 6.15) of the confidentiality constraint C. Let us suppose that the property encrypted (represented using Σ_{enc}) is provided over the object ob which allows us to deduce that:*

$$\Sigma \ \cup \ \Sigma_{enc} \ \models \ \exists k', ob_e : enc\_key(k') \wedge encrypted(ob, k', ob_e) \tag{6.17}$$

Then, using the https_post mechanism, the data owner will outsource ob_e to the csp. In fact, regarding that the encryption key k' is kept secret by the data owner, then the csp will not know any information about the key k'. This fact can be specified as follows:

$$\Box \ (\neg knows(s, k')) \tag{6.18}$$

Then, from Formula 6.18 and Hypothesis 1 (Section 5.6.1), we deduce the following:

$$\Box(\neg knows(s, ob)) \tag{6.19}$$

Finally, based on Formulas 6.8, 6.17, and 6.19 we deduce that:

$$\Sigma_{enc} \ \cup \ \Sigma \ \cup \ \Sigma_{m_{bgs}} \ \vdash \ \Sigma_C^1 \tag{6.20}$$

6.2.1 Fourth step: Choosing the Right Security Mechanisms.

The previous steps permit to select the *best goal satisfier* m_{bgs} that can satisfy the goal G, the corresponding set of violated constraints $\mathcal{C}$, and for each constraint $C_i \in \mathcal{C}$, the set of properties $\mathcal{P}_i$ that can satisfy C_i when applying m_{bgs}. At this level, based on those properties, the main goal is to select from the security mechanisms toolbox, the combinations that can *usefully satisfy* each violated constraint in $\mathcal{C}$.

Definition 27. ***(Useful satisfaction).*** *Given a violated security constraint C defined over an object ob, a set of security properties $\mathcal{P}$ that satisfy C, a set of utility constraint $\mathcal{U}_{Ob}$ defined over the object ob, and a set of formulas Σ representing the specification of a target system. A combination of mechanisms MC usefully satisfies the constraint C if the following condition holds:*

$$\Sigma \cup \Big\{ \bigwedge_{m \in MC} apply(m, ob) \Big\} \models \Big(\bigwedge_{P \in \mathcal{P}} P \quad \bigwedge_{u \in \mathcal{U}_{ob}} provides(u, ob) \Big) \tag{6.21}$$

Example 15. *We consider that we will use the same scenario as in Example 14, except in this example we consider that the data owner requires to be able to perform computation (CP) and order search (OS) over the information stored in the object to be outsourced ob, and (2) the mechanisms toolbox that can be used contains the security mechanisms described in Table 5.3. We have seen in Example 14 that the encryption property allows to enforce the confidentiality constraint C while the m_{bgs} (https_post) is applied. According to Formula 5.20, the encryption property can be provided by all encryption-based security mechanisms. However, not all of them can provide the required utility requirements (CP and OS). According to Table 5.3 Boldyreva [Boldyreva et al. 2009] security mechanism can provide the utility requirement OS, and Paillier [Paillier 1999] mechanism can provide the utility requirement CP. This can be formalized as follows:*

$$\begin{aligned} apply(Boldyreva, ob) \wedge apply(Paillier, ob) \rightarrow \big(\exists k, ob_e : enc\_key(k) \wedge \\ encrypted(ob, k, ob_e) \wedge provides(CP, ob) \wedge provides(OS, ob)\big) \end{aligned} \tag{6.22}$$

Then, by considering that $MC = \{Paillier, Boldyreva\}$, $\mathcal{P} = encrypted$, and $\mathcal{U}_{ob} = \{CP, OS\}$, we can formally deduce Formula 6.21 from Formula 6.22. Therefore, According to Definition 27, a combination of Boldyreva and Paillier encryption based mechanisms is a useful satisfier of the constraint C.

In this section, we have defined a four steps reasoning method for our formal model yield to pick up the set of security mechanisms that can enforce each security property required by the data owner. However, the reasoning method we proposed does not take into consideration conflicts that may occur between security mechanisms which makes enforcing a combination of security mechanisms that satisfies many security requirements hard to fulfill. We strive to overcome this limitation in the next section by extending and using a Graphplan-based approach to build a planning graph representing all possible transformations of the target system resulting from the application of the set of security mechanisms we got previously using our four steps reasoning method. Finally, we define a method to search the best security mechanisms execution plan to transform the used system from its initial state to a state in which the security requirements are enforced.

6.3 Security Mechanisms Planning to Enforce Security Policies

We strive to plan a sequence of mechanisms allowing to transform the system from its initial state to a state in which the goals are reached while respecting a set of defined constraints. Planning efficiency can be improved by allowing parallel application of mechanisms, which leads to minimize the number of parallel plan steps. In order to be able to apply mechanisms parallelly, we should figure out which mechanisms are compatible by finding different kind of conflicts between them.

Definition 28. ***Conflicting mechanisms.*** *Two mechanisms m_1 and m_2 represented respectively by $(\Delta_{m_1}, \Sigma_{m_1})$ and $(\Delta_{m_2}, \Sigma_{m_2})$, where Δ_{m_i} and Σ_{m_i} represent respectively the specifications of preconditions and the effects of m_i, are effectively incompatible if and only if one of the following deductions hold:*

1. $\Sigma_{m_1} \cup \Sigma_{m_2} \vdash \bot$.
2. $\Sigma_{m_i} \cup \Delta_{m_j} \vdash \bot$ *with* $1 \leq i, j \leq 2$ *and* $i \neq j$.
3. $\Delta_{m_1} \cup \Delta_{m_2} \vdash \bot$.

Item 1 means that the effects of m_1 and m_2 are inconsistent. Figure 6.1 illustrates this case by showing that in the case of $k-anonymity$ and $aes-cbc$, if an object ob is $k-anonymized$, we will lose the precision of the information stored in ob, as those information will be generalized, which is not the case when the $aes-cbc$ encryption mechanism is used. Obviously, we can see that we have a logical contradiction between the two mechanisms effects.

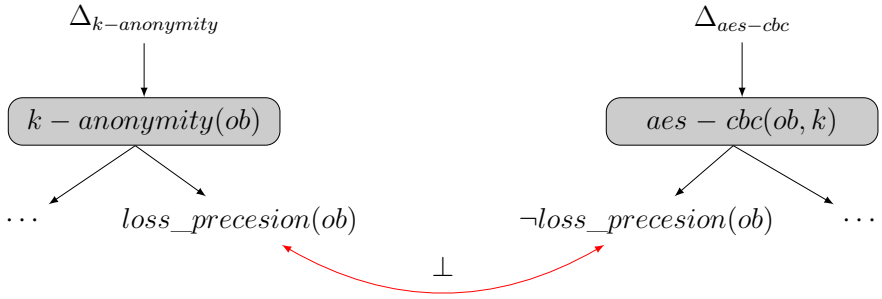


Figure 6.1 – Two conflicting mechanisms: The effects of $k-anonymity$ and $aes-cbc$ are inconsistent.

Item 2 of Definition 28 means that the effects of the application of m_i dissatisfy the preconditions of m_j. Figure 6.2 illustrates this case by taking the same mechanisms as

in Figure 6.1, except that we consider that an object can be $k-anonymized$ only if it is not encrypted (according to Formula 5.22). We can easily figure out the logical contradiction between the preconditions of $k-anonymity$ and the effects of $aes-cbc$.

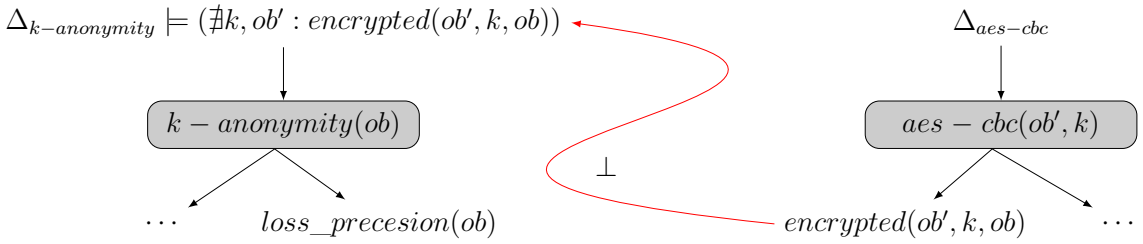


Figure 6.2 – Two conflicting mechanisms: The preconditions of $k-anonymity$ and the effects of $aes-cbc$ are inconsistent.

Item 3 of Definition 28 states that m_1 and m_2 have a competing preconditions such that they cannot be true in the same state of the target system.

Definition 29. ***(Parallel plan).*** *Consider a set of available mechanisms $\mathcal{M}$. A parallel plan is a finite sequence of sets of mechanisms $\mathcal{P} = \{p_1, \cdots, p_n\}$ such that:*

$$\forall p_i \in \mathcal{P} : p_i \subseteq \mathcal{M}.$$

Definition 30. ***(Correctness).*** *Given a system $\mathcal{S}$, its current state w_1, a finite set of mechanisms $\mathcal{M}$. A parallel plan $\mathcal{P} = \{p_1, \cdots, p_n\}$ is correct regarding $\mathcal{S}$ and w_1 if and only if the following conditions hold:*

1. $\exists w_2, \cdots, w_n$ *such that :* $\forall m \in p_i, w_i \models \Delta_m$, $1 \leq i \leq n$.
2. $\forall p_i \in \mathcal{P}, \forall m_1, m_2 \in p_i :$ m_1 *and* m_2 *are not conflicting (Definition 28).*

Problem 2. ***(Parallel Planning Problem).*** *Consider a system $\mathcal{S}$, its current state w_1, a set of mechanisms $\mathcal{M}$ that can be applied over $\mathcal{S}$, a set of goals $\mathcal{G}$ that should be achieved, and a set of constraints $\mathcal{C}$ that should be respected. The Parallel Planning Problem consists on finding a sequence of sets of mechanisms $\mathcal{P} = \{p_1, \cdots, p_n\}$ such that the following conditions holds:*

1. $\mathcal{P}$ *is correct regarding* $\mathcal{S}$ *and* w_1.
2. $\forall w_i = \Phi(w_{i-1}, p_{i-1}),\ \forall c \in \mathcal{C} :$ $w_i \models c$, $2 \leq i \leq n$.
3. $\forall G \in \mathcal{G} : w_{n+1} \models G$.

Informally speaking, item 2 in the previous problem states that, in any state w_i of $\mathcal{S}$ obtained by the application of the sets of mechanisms $p_1, \cdots, p_i$ $(2 \leq i \leq n)$, the set of security constraints $\mathcal{C}$ is satisfied. Item 3 ensures that the set of goals $\mathcal{G}$ is satisfied in the last state of $\mathcal{S}$ obtained by the application of $\mathcal{P}$.

In next part, we briefly introduce the Graphplan's basic operations as defined in [Blum and Furst 1995, Blum and Furst 1997]. Graphplan uses action schemata in the STRIPS format in witch each action is represented as preconditions and effects. This representation of action as preconditions and effects is suitable with the representation of our mechanisms parallel planning problem.

6.3.1 Graphplan Description

Graphplan is a directed, leveled graph composed of two kinds of nodes and tree kinds of edges. Graphplan levels alternate between *fact levels* containing *fact nodes* (each node is labeled with an instance of a predicate belonging to our formal language), and *action levels* composed of *action nodes* (each labeled with an instance of a security mechanism belonging to the mechanisms toolbox used in our model). Relations between actions and predicates in a Graphplan are explicitly represented through edges. *Preconditions-edges* are used to connect action nodes of an action level i to their preconditions in the fact level i. *Effects-edges* connect action nodes belonging to the action level i to their effects in the fact level $i + 1$. *Mutual-exclusion edges* are relations connecting action nodes belonging to the same Graphplan level. They represent conflicts between action nodes that can be identified according to Definition 28. Two action nodes belonging to the same action level and representing two instances of security mechanisms are in conflict according to Definition 28 means that no correct plan (Definition 30) could possibly contain both. Therefore, the use of mutual-exclusion edges is very useful to reduce the search space for a sub-graph of a Graphplan that might correspond to a correct plan.

Graphplan is based on two main phases: The first is called Graphplan construction phase consisting of growing a planning graph. The second phase allows to extract possible solutions (plans) from the planning graph by performing a backward searching phase starting with the goals. In the graph construction phase, we start with a planning graph having only a single fact level which contains the initial specification of the target system. GraphPlan construction method runs in stages, in each stage i, it extends the planning graph resulting from the stage $i - 1$ by adding one time step which contains the next action level and the following fact level. After each stage, Graphplan check if all predicates representing the goals are presented in the last fact level in the planning

graph, if it is the case, search a valid plan that transforms the system from its initial state to a state in which all the goals are achieved.

6.3.2 Graphplan Modelling of the Planning Problem

The STRIPS system [Fikes and Nilsson 1971] used by Graphplan is represented by four lists, a finite set $\mathcal{C}_s$ of ground atomic formulas called conditions, a finite set of operators $\mathcal{O}_s$ where each operator is composed of two formulas (satisfiable conjunction of conditions) representing its preconditions and effects, a finite set of instances of predicates $\mathcal{I}_s$ that denotes the initial state, and a finite set of instances of predicates $\mathcal{G}_s$ that denotes goal state. As we have seen in the previous section, our planning problem is composed of a system $\mathcal{S}$, a set of security mechanisms $\mathcal{M}$, a set of constraints $\mathcal{C}$, and a set of goals $\mathcal{G}$. Obviously, $\mathcal{S}$, $\mathcal{M}$, and $\mathcal{G}$ can be easily modeled as a STRIPS planning problem by expressing $\mathcal{S}$ as $\mathcal{C}_s$ and $\mathcal{I}_s$, $\mathcal{M}$ as $\mathcal{O}_s$, and $\mathcal{G}$ as $\mathcal{G}_s$. According to the security policy specification in section 5.5, $\mathcal{C}$ will be composed of security and utility constraints. Utility constraints specify the functionalities that should be provided for $\mathcal{S}$ (e.g. the ability to compare the equality of objects). A plan P satisfies a set of utility constraints C_u if at its end none of the utility constraints in C_u is violated. In other words, if one of the utility constraints is violated at some state of the plan, it must become true again by its end. Consequently, utility constraints will be expressed as goals in the STRIPS planning problem.

Security constraints specify the requirements that should be respected during the transformation of $\mathcal{S}$, they can be considered as safety constraint meaning that those requirements are to be satisfied in all states of $\mathcal{S}$. However, the STRIPS language as it is defined in [Fikes and Nilsson 1971] cannot express this kind of constraints. To overcome this limitation, we extend the STRIPS system language by adding the operator *Constraint* allowing to express the set of security constraints. For instance, a confidentiality constraint (rule 5.7) is to be expressed as follows:

Constraint ($\textbf{confidentiality\_constraint}_1(ob, e)$:

Formula: $sensitive(ob) \wedge untrusted(e) \wedge knows(e, ob)$)

In the previous expression, *confidentiality_constraint*$_1$ *(o,e)* is used to denote the name of the constraint and the variables (bounded variables of the rule 5.7) to be instantiated. The *Formula* field represents the conjunction of predicates indicating the condition under which the constraint is violated (the negation of CNF representation of the constraint). For instance, according to the previous expression, *confidentiality_constraint*$_1$ (ob, e) is violated if there exists a sensitive object *ob* that is known by an untrusted entity e.

6.3.3 Extending Planning Graph: Planning Under Security Constraints

Graph Construction Phase Extension

We extend Graphplan's construction method of the planning graph in two ways. The first extension allows to build a planning graph of a planning problem which contains *Domain Axioms* (axioms that formally specify relations between different objects of the system). Second, we improve the Graphplan's construction method of the planning graph to avoid the violation of security constraints while building the planning graph.

The need of axioms. In Graphplan approach, the lack of axioms disrupts the ability to represent real-word domains, which contains normally quite complex conditions and rules. Without the use of axioms, mechanisms preconditions and effects can become quickly too complex and unreadable. In our approach, we believe that the use of axiom will provide a natural way of deriving supervenient properties, which represents logical consequences of the effects of applied mechanisms.

Example 16. *Suppose that we have two mechanisms in the mechanism toolbox to be used:* $aes-cbc$ *and http_post. The effects of http_post are represented by Formula 6.2 (Example 11) and the effects of* $aes-cbc$ *are described as following:*

$$\Sigma_{aes-cbc} = \{\exists k, ob_e : enc\_key(k) \wedge encrypted(ob, k, ob_e)\} \tag{6.23}$$

Let us suppose that the parallel plan $\mathcal{P} = \{aes-cbc(obj, k), \{http\_post(obj_e, u), http\_post(k, u)\}\}$ *is performed over* $\mathcal{S}$ *and transforms it to the state* w *such that:* $w \models encrypted(obj, k, obj_e) \wedge knows(u, obj_e) \wedge knows(u, k)$*. According to Hypothesis 1 (Section 5.6.1), we know that if an entity* e *knows* obj_e *and the used encryption key* k*, then it can know* obj*. Unfortunately, Hypothesis 1 is difficult to be expressed using only the two mechanisms encrypt and send. This lack of expressiveness can be overcame by using Hypothesis 1 as an axiom in the target system.*

It is clear that the use of axioms can add an important expressive power to the planning problem specification. However, the Graphplan construction method should be improved to be able to use axioms to infer new facts about the target system.

Updating Knowledge Using an Inference Graph

In this part, we present how we define axioms in our approach and the way they will be used in the planning process. Axioms have been defined by [Ghallab et al.] as "log-

ical formulas that assert relationships among propositions that hold within situation". According to this definition, we define an axiom as an expression in the following form:

$$\bigwedge_{i=1}^{n} p_i \rightarrow \bigwedge_{j=1}^{m} q_j \tag{6.24}$$

Where p_i and q_j are instances of some predicates of our defined language.

According to a state w of the target system, we want to be able to infer all possible new facts based on a set of axioms that represents relationships between different predicates in our language. To meet this goal, we utilize the same construction method used in Graphplan in order to build an inference graph. In this construction method, we consider each axiom in our system as an action, then the left part of the representation of the axiom (6.24) will be considered as the preconditions of the action and the right part is its effects. The idea consists on applying in each layer of the inference graph the set of applicable actions (axioms) until we infer all possible new facts (we find the same set of facts in the last two Graphplan's fact-layers). Algorithm 7 describes how the inference graph is constructed.

Once the inference graph is built, it allows to extract the set of facts that are derived using the set of defined axioms. In fact, the set of inferred facts is $\mathcal{IG}_l \backslash \mathcal{IG}_0$ were $\mathcal{IG}_0$ and $\mathcal{IG}_l$ represent respectively the set of predicate-nodes in the first fact level and the set of predicate-nodes in the last fact level of the inference graph.

Theorem 6. *Given the set of formulas Σ_w representing the system $\mathcal{S}$ in the state w, and a set of n consistent axioms $\mathcal{A} = \{ax_1, \cdots, ax_n\}$, the height of the graph representing the inference graph of $\mathcal{S}$ using $\mathcal{A}$ will be at most n.*

Proof. Since axioms are used to deduce new relations between objects and cannot create new objects in $\mathcal{S}$, and that $\mathcal{S}$ is composed of a finite set of objects, we can deduce that the inference graph will be built in finite time. Furthermore, as we will use Graphplan construction method to build the inference graph, we will be able to parallelly use axioms to infer new facts and then to reduce the height of the inference graph. The worst case will be when all axioms cannot be applied on the system parallelly (the applicability of ax_i depends on the facts derived by ax_{i-1}), which clearly requires an n-level inference graph to deduce all new facts from $\mathcal{A}$. □

Theorem 7. *Consider a system $\mathcal{S}$ composed of n objects and represented by p predicates in a state w_i, and m axioms each having a constant number of bounded variables. Let q be the largest number of predicates in the right-side of each axiom (formula 6.24) and v be the largest number of bounded variables in any axiom. Then, the size complexity of a k-level inference graph created using Graphplan construction method and the time complexity of building the graph, are polynomial n, m, q, p and k.*

input :
$\mathcal{G}$ /* planning graph (Graphplan) */
$last\_fl_G = \{f_1, \cdots, f_n\}$ /* set of facts in the last fact level of G */
$\mathcal{A}_x = \{Ax_1, \cdots, Ax_m\}$ /* the set of domain-axioms */
output:
$inferred\_facts$ /* the set of derived new facts */

Main
$\mathcal{IG} = \emptyset$ /* inference graph initialization */
$add\_fact\_level(\mathcal{IG}, last\_fl)$ /* add the last fact level of $\mathcal{G}$ to the inference graph $\mathcal{IG}$ */
for $i = 0$ ***to*** m **do**
  $new\_fact\_level = \emptyset$ /* new empty fact level */
  $new\_fact\_level = last\_level(\mathcal{IG})$ /* copy the facts in the last fact level of $\mathcal{IG}$ to new_fact_level */
  foreach $axiom$ ***in*** $\mathcal{A}_x$ **do**
    $instances = instantiate(axiom)$ /* get all instances of the $axiom$ */
    foreach $inst$ ***in*** $instances$ **do**
      /* axioms can be divided into left and right parts (rule 6.24) */
      if $(last\_level(\mathcal{IG}) \models left\_part(inst))$ **then**
        $new\_fact\_level = new\_fact\_level \cup right\_part(inst)$
      end
    endfch
  endfch
  if $(new\_fact\_level == last\_level(\mathcal{IG}))$ **then**
    $inferred\_facts = new\_fact\_level \setminus last\_fl_G$
    break
  else
    $add\_fact\_level(\mathcal{IG}, new\_fact\_level)$
  end
end

Algorithm 7: Building inference graph and getting new derived facts

Proof. Since axioms cannot create new objects, the maximum number of predicates that may be created during the instantiation of an axiom is qn^v. Therefore, in any fact level of the inference graph, the maximum number of nodes is $p + mqn^v$. The maximum number of nodes in any axiom-level (action level) of the inference graph is mn^v since any axiom can be instantiated in at most n^v distinct ways in worst case. Therefore, the size complexity of a k-level inference graph created using Graphplan

construction method is $O(k(p+mqn^v+mn^v))$. As a result, for a fixed v (in our formal model $v = 6$), the size complexity of a k-level inference graph is polynomial n, m, q, p and k. In the other side, the complexity time of creating new inference graph level (axiom level and fact level) is $O(mn^v)$. Therefore, the complexity time of creating a k-level inference graph is $O(kmn^v)$. As a result, for a fixed v, the time complexity of creating a k-level inference graph is polynomial n, m, and k. □

Building Planning Graph Under Security Constraints

The specification of security constraints requires that some properties should be respected during all the states of the target system. Since each fact level of the planning graph is built using the construction method of Graphplan, it can be considered as a possible state of the system, our idea consists of verifying the satisfiability of security constraints on each new created fact level of the planning graph during its construction.

Definition 31. ***(Violated security constraint).*** *Consider a planning graph G composed of n fact levels $fl_1, \cdots, fl_n$, each fact level fl_i is composed of a set of facts w_i. A security constraint C specified in our formal language using the set of formulas Σ_C and specified in the STRIPS system language by $\overline{\Sigma_C}$ (the negation of CNF representation of Σ_C) is violated in a fact level fl_i if and only if $w_i \models \overline{\Sigma_C}$.*

Graphplan uses directed edges to connect each action instance node belonging to the ith action level of the graph to the set of fact nodes belonging to the ith fact level representing its preconditions, and to the set of fact nodes belonging to the $(i+1)$th fact level representing its effects. Thanks to this property, we are able to find the combinations of actions belonging to the ith action level of the graph and leading to security constraints violation in the $(i+1)$th fact level.

Algorithm 8 describes the used method to get the combinations of actions leading to violate a security constraint. The correctness and the complexity of the Algorithm 8 are proved by the following theorems.

Theorem 8. ***(Correctness).*** *Given a violated security constraint C and a set of fact nodes cause_nodes that causes the violation of C, the Algorithm 8 terminates and computes all the combinations of actions that lead to violate C.*

```
   input :
           C   /* the violated security constraint */
           cause_nodes   /* the set of fact_nodes that causes the violation of C
   */
   output:
           action_combinations    /* the set of combinations of actions that
   violates the constraint C */
 1 Main
 2 combination = ∅
 3 all_combinations(causes_nodes, action_combination)
 4 End Main
 5
 6 Recursive Procedure all_combination(nodes, combination)
 7 if (Card(nodes) == 0) then
 8 |   add(combination, action_combination) /* add combination to
   |   action_combination */
 9 end
10 first_node = nodes.first    /* get the first node in the set nodes */
11 remove(nodes, first_nodes)    /* remove the first_node from the set nodes */
12 foreach action_node in first_nodes.in_edges do
13 |   /* first_nodes.in_edges represents the set of edges connecting the node
   |   first_nodes to the actions that provide it. */
14 |   copy_combination = combination
15 |   if (action_node ∉ copy_combination) then
16 |   |   add(action_node, copy_combination)
17 |   end
18 |   all_combinations(causes_nodes, copy_combination)
19 endfch
```

Algorithm 8: Getting all combinations of actions that lead to violate a constraint

Proof. To prove the correctness of the algorithm 8, we have to show that (1) it terminates; (2) it computes all the combinations of actions that lead to the violation of C.

1. Algorithm 8 contains the recursive procedure "all combination" having the termination case in which the set of fact nodes *nodes* is empty (line 7). Since the number of fact nodes that cause the violation of C is supposed to be finite (the cardinality of *cause_nodes* is finite), the set of actions that provides each fact node in *cause_nodes* is finite, and that the recursive procedure removes an ele-

ment from *cause_nodes* (line 11) during each new recursive call, we can deduce that algorithm 8 terminates .

2. When the procedure "all combination" is called for the first time (line 3), the set of fact nodes causing the violation of the constraint C and an empty combination of actions are passed as parameters. The first node in *cause_nodes* is selected and removed from the list. Then, for each action node *action_node* allowing to provide the selected node (each parent node in the graph), we make a new copy *copy_combination* of *combination*, insert *action_node* into *copy_combination*, and call recursively the procedure using the *cause_nodes* and *copy_combination*. When a recursive call reaches the termination case (line 7), we are sure to have for each fact node in *cause_nodes*, an action node in *combination* that provides it. Moreover, since we have a recursive call for each action node providing a fact node in *cause_nodes*, obviously we will get all possible combinations of actions leading to violate the constraint C.

□

Theorem 9. ***(Complexity).*** *Given a violated security constraint C, a set of cause nodes* $CN = \{n_1, \cdots, n_n\}$ *representing the set of fact nodes that causes the violation of C, the complexity of the algorithm 8 is* $O(\prod_{i=1}^{n} l_i)$ *in time, where* l_i *is the number of different actions providing the fact node* n_i*.*

Proof. Since, each call of the procedure "all combination" will select a fact node $n_i \in CN$ and will create l_i recursive calls, the total number of recursive calls will be $\prod_{i=1}^{n} l_i$. Therefore, the complexity of the algorithm 8 is $O(\prod_{i=1}^{n} l_i)$ in time. □

Once we know the combination of actions $\mathbb{C}_c$ that leads to the violation of the security constraint C. The trivial way to solve this violation problem would be to remove $\mathbb{C}_c$ and its corresponding effects from the planning graph. Unfortunately, this solution can be useless in many cases as it can prevent some actions in $\mathbb{C}_c$ (a subset of $\mathbb{C}_c$) that do not violate C to be used.

Avoiding security constraints violation. In Graphplan, mutual exclusions are basically used to specify that no valid plan could possibly contain conflictual actions in the same plan step. Since, a security constraint C is violated if all actions in a combination $\mathbb{C}_c$ that violates C are applied in the same action level of the planning graph, our solution to prevent this violation is to use mutual exclusion relations as following:

1. If $|\mathbb{C}_c| \geq 2$: $\forall\ node_a \in \mathbb{C}_c$, create a mutual-exclusion between $node_a$ and $\mathbb{C}_c \setminus \{node_a\}$.
2. If $|\mathbb{C}_c| = 1$, remove the action-node in $\mathbb{C}_c$ and its corresponding effects from the planning graph .

where $|\mathbb{C}_c|$ represents the number of action-nodes in $\mathbb{C}_c$. Condition 1 ensures that if the number of action-nodes in $\mathbb{C}_c$ is more that one, therefore we will create a mutual-exclusion between each action-node $node_a$ in $\mathbb{C}_c$ and the set of other action-nodes in $\mathbb{C}_c$. This allows in one side to ensure that no correct plan could possibly contain $node_a$ and $\mathbb{C}_c \setminus \{node_a\}$ together which allows to avoid the violation of the security constraint C, and on the other side to allow the largest subsets of action-nodes ($\mathbb{C}_c \setminus \{node_a\}$) in $\mathbb{C}_c$ that do not violate C to be used together in the same plan. Condition 2 states that if $\mathbb{C}_c$ is composed of only one action-node, therefore, the unique solution to avoid the violation of C is to remove the action-node in $\mathbb{C}_c$ as well as its corresponding effects from the planning graph.

Example 17. *Consider a system $\mathcal{S}$ composed of three entities, a data owner, a user u and a server s. The data owner wants to outsource a sensitive object o to a server s in order to share its content with the user u. The data owner considers that s cannot be fully trusted, therefore the confidentiality of o must be protected. The specification of the system $\mathcal{S}$ is as follows:*

- *Goal:* $outsource(o, s)$.
- *Constraint:* $\Box\ \neg knows(s, o)$
- *Axioms:*
  - Ax_1 : *Formula 5.21 (Section 5.6.1)*
  - $Ax_2 : \forall o, o_{enc}, k, s.\ \ csp(s) \wedge encrypted(o, k, o_{enc}) \wedge knows(s, o_{enc}) \rightarrow outsource(o, s)$

Axiom Ax_1 specifies that if an entity e knows o_{enc} (o_{enc} represents the encrypted form of o using the key k) and the encryption key k, then e knows o. Axiom Ax_2 specifies that if a cloud storage server s knows an object o_{enc} representing the encrypted form of o (o_{enc} is sent to s) therefore we can consider that o is outsourced.

Figure 6.3 presents a subgraph of the planning graph constructed for $\mathcal{S}$. The *no_op* action is used to include the facts of the fact level i into the fact level $i + 1$. As we can see in the subgraph, the application of the two instances $http\_post(k_1, s)$ and

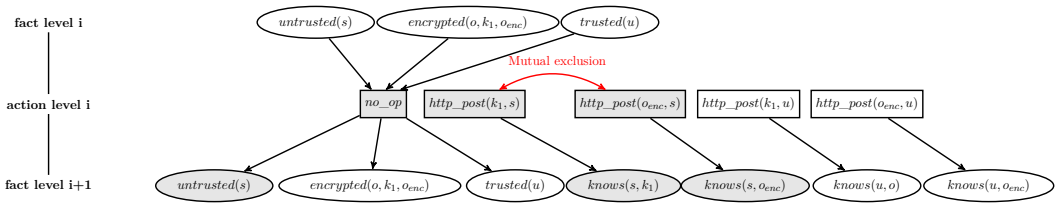


Figure 6.3 – Subgraph of the planning graph constructed for Example 17

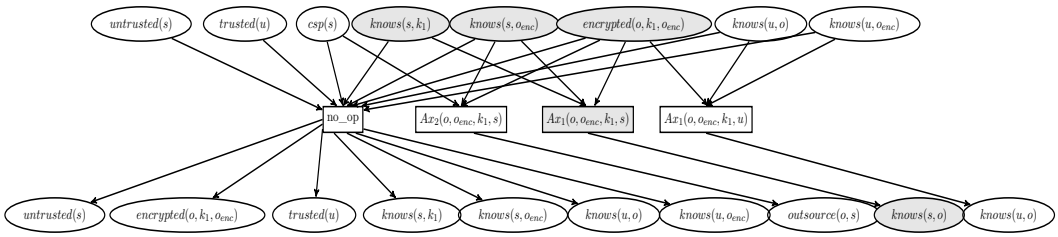


Figure 6.4 – The inference graph used to infer new facts from the level $i + 1$ of the planning graph in Figure 6.3

$http\_post(o_{enc}, s)$ of the action $http\_post$ will include two new facts $knows(s, k_1)$ and $knows(s, o_{enc})$ into the fact level $i + 1$. Now, based on the inference graph (Figure 6.4), and using the instance $Ax_1(o, o_{enc}, k_1, s)$ of the axiom Ax_1 we deduce $knows(s, o)$ which violates the confidentiality constraint. A bottom up analysis of the two subgraphs starting from $knows(s, o)$ allows to conclude that the combination of actions that violates the constraint is composed of $http\_post(k_1, s)$ and $http\_post(o_{enc}, s)$ (we exclude the no_op action as it should be always applied). A mutual exclusion will be specified between $http\_post(k_1, s)$ and $http\_post(o_{enc}, s)$ to exclude all plans in which the two actions are simultaneously applied in the ith plan step.

6.3.4 Searching the Best Security Mechanisms Plan

Given a planning graph $\mathcal{G}$ constructed using our previously explained extension of GraphPlan, our goal is to find the best mechanisms execution plan (parallel plan) that enforces the chosen security and utility requirements. For this end, and to be able to compare different mechanisms execution plans, as a first step, we assign a weight for each action-node in $\mathcal{G}$ representing a security mechanism using the metric described in Definition 32. As a second step, we define a second metric to measure a score for each mechanisms execution plan that can satisfy the defined policy as described in Definition 33.

Definition 32. *Consider an action-node an_m in $\mathcal{G}$ representing the application of an instance of the security mechanism m over the object ob. Suppose that m provides*

n security properties $sp_1, \cdots, sp_n$ *and p utility properties* $up_1, \cdots, up_p$*. The weight* ω *which will be assigned to* an_m *is measured as following:*

$$\omega = \alpha_{ob} \sum_{i=1}^{n} \tau_i + \beta_{ob} \sum_{i=1}^{p} \nu_i - \delta_{ob}\ \varepsilon_m$$

where $\tau_i \in [0,1]$ *represents the robustness level of the provided security property* sp_i*,* $\nu_i \in [0,1]$ *represents the satisfiability level of the provided utility property* up_i*,* $\varepsilon_m \in [0,1]$ *is the deployment efficiency level of the mechanism* m*, and* $\alpha_{ob} \in [0,1]$*,* $\beta_{ob} \in [0,1]$*, and* $\delta_{ob} \in [0,1]$ *represents respectively the security, utility, and deployment efficiency factors of ob such that* α_{ob}*,* β_{ob}*, and* δ_{ob} *are complementary.*

The intuitions behind the use of the robustness level τ (1), the satisfiability level ν (2), the deployment efficiency level ε (3), and the security, utility and deployment efficiency factors (4) to measure the weight of an action-node is that:

1. Some security mechanisms are not as robust as they should be to fully ensure their provided security properties under well known attacks. For example, encryption-based mechanisms are supposed to ensure the confidentiality of the objects over which they are applied. However an Order-preserving encryption based mechanisms such as Boldyreva [Boldyreva et al. 2009] preserves the order of the plaintexts, which may enable many attacks. It was concluded that order-preserving encryption leaks at least half of the plaintexts bits [Xiao and Yen 2012]. Hence, the confidentiality robustness level $\tau_{confidentiality}$ will be less than 0.5 for Boldyreva.

2. Some security mechanisms cannot fully provide some utility requirements. In these cases, the satisfiability level factor ν is used to specify the level of providability of an utility requirement. For illustrative purpose, let us take the example of homomorphic-based encryption mechanisms which are supposed to provide computation (addition + multiplication) over encrypted objects. However, Paillier cryptosystem [Paillier 1999] is an homomorphic-based encryption mechanism allowing to perform only addition over encrypted data. Therefore, satisfiability level factor of computation for Paillier cryptosystem will be $\nu_{computation} = 0.5$.

3. Some security mechanisms are expensive in terms of deployment time compared to other security mechanisms, we take this fact into consideration by using the deployment efficiency level ε_m, as much as the mechanism m can be efficiently deployed, ε_m will be closer to 1.

4. The weight of an_m representing the application of m over *ob* should also take into account the security, utility and deployment efficiency factors represented

respectively by ϵ_{ob}, ρ_{ob}, and δ_{ob}, which are specified by the data owner for the data object ob. For illustrative purpose, let us take a file f_1 storing information about the payment parameters used by the costumers of a company. The company attributes the value 0.8 to ϵ_{f_1}, 0.1 to ρ_{f_1}, and 0.1 to δ_{ob} as it considers that the utility of f_1 as well as deployment efficiency of the policy over f_1 are not important compared to its security. As a result, the action-node in $\mathcal{G}$ representing a security mechanism applied over f_1 which ensures the highest level of robustness for security properties will have the highest weight compared to others having high providability of utility requirements, high deployment efficiency and weakly robustness for security properties.

Definition 33. *Consider a parallel plan $\mathcal{P} = \{p_1, \cdots, p_n\}$. Suppose that each $p_i \in \mathcal{P}$ is composed of l_i action-nodes $an_1^i, \cdots, an_{l_i}^i$. The score Sc of $\mathcal{P}$ is:*

$$Sc = \sum_{i=1}^{n} \sum_{j=1}^{l_i} \omega_j^i$$

where ω_j^i is the weight of the action-node an_j^i measured according to the Definition 32.

Definition 34. *Consider a security policy $\mathcal{SP}$ and a set of parallel plans $\mathcal{P}_1, \cdots, \mathcal{P}_n$ in $\mathcal{G}$ each satisfying $\mathcal{SP}$ and all having respectively the scores $Sc_1, \cdots, Sc_n$. A parallel plan $\mathcal{P}$ having the score Sc is the best parallel plan in $\{\mathcal{P}_1, \cdots, \mathcal{P}_n\}$ if the following condition holds:*

$$\forall i \in 1 \cdots n,\ Sc \ \geq \ Sc_i$$

Obliviously, finding the best parallel plan in a planning graph $\mathcal{G}$ that enforces a security policy $\mathcal{SP}$ requires finding all parallel plans in $\mathcal{G}$ that satisfy $\mathcal{SP}$. Unfortunately, the computation of all parallel plans in $\mathcal{G}$ that satisfy $\mathcal{SP}$ is NP-hard, as we will prove in the following.

Theorem 10. *Computing all parallel plans in a planning graph that enforce a security policy $\mathcal{SP}$ is NP-hard.*

Proof. We prove the previous result by showing that finding all constrained paths that connect two nodes in a graph is NP-hard. If it is not, one could determine whether a graph has a Hamilton path or not by checking whether a path exists with length $k-1$ (k is the number of nodes in the graph). This problem, however, is known to be NP-complete [Bertossi 1981]. Therefore, computing all parallel plans in a planning graph that enforce a security policy is NP-hard. □

Since the problem of computing all parallel plans in a planning graph that enforce a security policy is NP-hard, we cannot expect to be able to resolve an arbitrary size instance of the problem of finding the best parallel plan that enforce a policy to optimality. Thus, heuristic resolution strategies are widely exploited to solve such a problem with a reasonable computational effort.

Heuristic search based planning

Our goal is to find the parallel plan having both the maximum score regarding our metric (defined in Definition 32, 33, and 34), and the minimum number of steps. To this end, we use the cost-optimal planner CO-PLAN [Robinson et al. 2008] which proceeds in four stages:

- Planning graph conversion to CNF wff: Convert the planning graph into a CNF notation by constructing proposition formula as described in [Kautz and Selman 1996].
- Wff solving: CO-PLAN uses a modified version of RSAT [Pipatsrisawat and Darwiche 2007] called CORSAT to process the CNF formulae which allows to figure out: (1) If a solution exists for the given decision problem, and (2) if a solution exists, it is identified with minimal plan costs.
- Bounded forward-search: CO-PLAN uses the speed and efficiency of SAT-based planners allowing to obtain a good admissible initial bound on the cost of an optimal plan. In the second phase, CO-PLAN performs then a bounded forward-search in the problem state space.
- Plan extraction: If a model of the wff is found, then the model is converted to the corresponding plan; otherwise, the length of planing graph is incremented and the process repeats.

In fact, CO-PLAN identify the solution having the minimal parallel plan costs. To be able to use it, we transform our parallel plan score maximization problem to a minimization plan cost problem by considering that $Cost_{\mathcal{P}} = -Sc_{\mathcal{P}}$, where $Cost_{\mathcal{P}}$ and $Sc_{\mathcal{P}}$ represent respectively the cost of the parallel plan $\mathcal{P}$ and the score of $\mathcal{P}$ measured according to Definition 33.

6.4 Implementation and Evaluations

In the experimental part of this work, we measure the computational performance of our approach.

6.4.1 Implementation

We develop a prototype implementing our approach to find a near-optimal security mechanisms plan allowing to enforce security policies for outsourced data using available open source C++ libraries. For GraphPlan construction, we used the SATPLAN'06 library [Kautz et al. 2006] allowing to create a planning graph up to some length k. We extend SATPLAN'06 library (as described in section 6.3.3) to support: (1) the use of domain axioms allowing to deduce new facts about objects of the system to be used, and (2) we improve the Graphplan's construction method of the planning graph to avoid the violation of security constraints while building the planning graph. For analyzing the planning graph and searching the best mechanisms plan, we used CO-PLAN library [Robinson et al. 2008].

6.4.2 Experimental Setup

As we are interested on planning security mechanisms to protect outsourced data, the domain that we have used in evaluating our prototype is composed of:

- A data owner;
- A finite set of users:
  - Trusted users: which can access and use the outsourced data
  - Untrusted users: which are not supposed to be able to violate the policy. In all experiments, we suppose that we have two untrusted users, a cloud storage server and an external adversary.
- A finite set of objects that represents the data to be outsourced, we consider that the data owner wants to outsource a file system. So the objects are the set of files and directories in the file system to be outsourced.
- A finite set of security and utility requirements representing the policy to be enforced. We suppose that the data owner will specify some security constraints and utility goals over some objects belonging to the file system to be outsourced. Only

the objects over which the data owner has specified the policy will be considered in the planning problem.

- A finite set of security mechanisms that can be used to enforce the security policy. In our prototype, we specified 20 security mechanisms, including 8 encryption-based mechanisms, 4 anonymization-based mechanisms, 6 watermarking-based mechanisms, and 2 information transfer protocols HTTPS and SSH that can be used to send the objects to be outsourced to the cloud server.

Appendix B describe the STRIPS specification of the target system we have used to evaluate our prototype. We ran the all experiments on a server with Intel core i7 2.50 GHz, 16 GB of RAM, and running Debian 7.

6.4.3 Experimental Results

We conducted a set of experiments to evaluate the performance of our prototype. Table 6.1 shows the parameters used in each experiment, the number of nodes in the planning graph built to resolve the problem, and the time needed to find a near-optimal solution using the method we presented in Section 6.3.4.

Parameters				**number of nodes**	**time(s)**
objects	**constraints**	**users**	**mecanisms**		
5	5	5	15	75952	1.9
10	10	5	15	121385	9.7
20	15	5	15	721385	97.65
100	50	5	20	1951423	721.5

Table 6.1 – Our prototype performance with respect to: the number of objects that will be outsourced (objects), the number of constraints defined over the object to be outsourced (constraints), the number of users involved in the used system (users), the number of security mechanisms that can be used to enforce the policy (mechanisms). Column "number of nodes" indicates the number of nodes in the created planning graph.

6.5 Conclusion and Contribution

One of the greatest challenge in this thesis was to define an efficient formal reasoning method allowing to use the formal model presented in Chapter 5 (i.e., the specification

of the target system, the policy to be applied over it, and existing security mechanisms) to figure out the combination of security mechanisms that can enforce the chosen policy over the target system. To meet this goal, as a first time, we have defined a four steps reasoning method for our formal model to pick up the set of security mechanisms that can enforce each security property required by the data owner. Then, as a second step, we extend and use a Graphplan-based approach to build a planning graph representing all possible transformations of the system resulting from the application of the set of security mechanisms we got previously using our four steps reasoning method. Finally, we define a method to search the near-optimal security mechanisms execution plan to transform the target system from its initial state to a state in which the security and utility requirements are enforced while offering the best trade-off between security, utility and complexity.

In our formal model, policies to be applied over the data to be outsourced are composed of a set of imperative goals (i.e., goals and utility requirements) and planning constraints (i.e., security constraints). Unfortunately, those policies are either wholly satisfied or violated, which allows our reasoning method only to be efficient when dealing with limited-scale policy enforcement over outsourced data problems. In the next chapter, we strive to overcome this limitation by designing a method allowing, in the case in which no mechanisms execution plan could fully satisfy the chosen policy, to choose the best compromise between the defined security constraints and the set of goals that should be satisfied.

CHAPTER 7

Best effort based Approach for Security Mechanisms Planning to Enforce Security Policies Over Outsourced Data

Humans make mistakes. When dynamic systems are controlled and managed by humans, the rate and consequences of those mistakes increase. Whether it occurs while controlling industrial machinery, managing sensitive data, or deploying security policies, human mistakes can lead to costly consequences. Automatic policy satisfaction and deployment need often to plan complex set of mechanisms to reach a set of fixed goals while ensuring a set of security constraints.

Actually, we have seen that in many real word planning scenarios (e.g., automatic security policy deployment), no mechanisms execution plan could satisfy the defined goals without violating some specified security constraints. In this chapter, we strive to design an approach allowing outsourced data owner to choose the best compromise between security constraints to be enforced and the set of goals to satisfy over the outsourced data. To this end, we extend the planning graph based approach presented in [Kautz and Selman 1999] by using a data tainting based method allowing to (1) mark the set of fact nodes that violate security constraints and (2) effectively propagate those taints to the fact nodes representing the goals that need to be satisfied. Later on, based on the propagated taints, we define a reasoning method allowing to get the near optimal compromise between the goals to satisfy and the security constraints to ensure.

The reminder of this chapter is organized as following. Section 7.1 discusses related work. Section 7.2 introduces preliminary concepts of the parallel planning under security constraints problem. Section 7.3 depicts our planning graph tainting approach as well as its efficient taint propagation methods. Section 7.4 presents our solution to find the security mechanisms execution plan that provides the best compromise between satisfying specified goals and ensuring security requirements. Section 7.5 reports the conclusion of this chapter. Finally, please refer to Appendix C for detailed proofs of the lemmas and theorems used in this chapter.

7.1 Related Work

The problem of security mechanisms planning to enforce security policies while attempting to satisfy a set of goals can be seen as the problem of actions planning to satisfy a set of goals while ensuring a set of safety constraints. This last problem have been investigated by the AI community. Frazzoli et al. [Frazzoli et al. 2000] proposed an algorithm that enables a system (i.e., a robot) to move from its original state to a new state (e.g., to accomplish an assigned task such as performing an observation or delivering a payload), while avoiding to violate a set of constraints (e.g., collisions with fixed or moving obstacles). They introduced the notion of τ-safety which indicates that a plan is safe (does not violate any safety constraint) for at least τ plan steps. Kindel et al. [Kindel et al. 2000] proposed a randomized motion planner for kinodynamic asteroid avoidance problem. In this problem a robot should avoid to collude with moving obstacles under kinematic, dynamic constraints while attempting to reach a specified goal state. The authors introduced the notion of an "escape trajectory" as a contingency plan which is to be taken in case the planner fails to find a path that satisfy the intended goals. Yoo et al. [Yoo et al. 2013] propose algorithms for model checking and policy synthesis that provide, for a given safety policy, a probabilistic quantitative measure of safety and completion time, and synthesize policies that minimize completion time with respect to a given safety threshold. Despite that the work in [Yoo et al. 2013] is interesting, it suffers form a major limitation as it based on the temporal logic called PCTL (Probabilistic Computation Tree Logic) which can only provide a plan solution that maximize the probability to reach a set of goals.

Data tainting is not a new concept. It is a mechanism to monitor and track how a specific information is propagated in a system. The main idea of this mechanism is to assign a set of tags to some of the data object in the target system and then spread those tags to other related objects to this data according to the evolution of the used

system. It is used mainly for vulnerability detection, protection of sensitive data, and more recently, for analysis of binary malware.

To detect vulnerabilities on PHP applications, Huang et al. [Huang et al. 2004] provided an algorithm relying on a lattice-based analysis derived from type systems and typestate, then compared it to a technique based on bounded model checking. Xie and Aiken [Xie and Aiken 2006] proposed an approach for detecting SQL injection vulnerabilities in PHP scripts. Jovanovic et al. [Jovanovic et al. 2010] tackle the problem of vulnerable web applications by means of static source code analysis. The propose a solution relying on the use of flow-sensitive, interprocedural and context-sensitive data flow analysis to discover vulnerable points in a program. Later on, Choi et al. [Choi et al. 2015] adopt taint analysis to provide binary analyzer that can find vulnerabilities and self-modifying code.

With private data protection in mind, data tainting analysis was used by Chan el al. [Chan et al. 2012] proposed DroidChecker that uses inter-procedural Control-Flow Graph (CFG) analysis and data taint checking to detect exploitable data paths in an Android application. Other approaches use dynamic data tainting to find privacy leaks. For instance, TaintDroid [Enck et al. 2010] provides an Android's virtualized execution environment to monitor Android applications during runtime and track how application leaks private information.

To analyze binary malware, recently, Wang and Shieh [Wang and Shieh 2015] define DROIT, a taint tracker that is able to dynamically alternate between object-level and instruction-level, which allows authors to successfully come up with a malware behavior profiling tool.

7.2 Preliminaries

In this section, we formally define the parallel planning under security constraint problem.

Definition 35. *An n-layered graph is a graph* $\mathcal{G} = (L = \{l_1, \cdots, l_n\}, E = \{e_1, \cdots, e_{n-1}\})$ *where the* l_i*'s are sets of vertices, and the* l_i *vertices are connected only to the* l_{i+1} *vertices through the edges* e_i*.*

Definition 36. ***(Planning Graph).*** *Given a system* $\mathcal{S}$*, a set of ground literals* w *representing the initial state of* $\mathcal{S}$*, and a finite set of mechanisms* $\mathcal{M}$*. The planning graph* $\mathcal{PG} = (FL, AL, E, ME)$ *of length* n *constructed for* $\mathcal{S}$ *using* $\mathcal{M}$ *is a (2n-1)-layered graph* $\mathcal{G} = (L = \{l_1, \cdots, l_{2n-1}\}, E = \{e_1, \cdots, e_{2n-2}\})$ *where the following conditions hold:*

1. $FL = \{fl_1, \cdots, fl_n\}$ *where* $fl_i = l_{2i-1}$;
2. $AL = \{al_1, \cdots, al_{n-1}\}$ *where* $al_i = l_{2i}$;
3. *Each ground literal in* w *is represented with a vertex in* fl_1;
4. $\forall l \in AL, \forall an \in l :$ *an is an instance of an* $m \in \mathcal{M}$;
5. $\forall l_j \in AL, \forall an \in l_j : \ \Delta_{an} \subseteq l_{j-1}$ *and* $\Sigma_{an} \subseteq l_{j+1}$;
6. $\forall l_j \in AL, \forall an \in l_j, \forall fn \in \Delta_{an} : \exists e \in e_{j-1}$ *that links* fn *to an;*
7. $\forall l_j \in AL, \forall an \in l_j, \forall fn \in \Sigma_{an} : \exists e \in e_j$ *that links an to* fn;
8. $ME = \{me_1, \cdots, me_{n-1}\}$ *where* $me_i = \{me | me = (an_j, an_k), j \neq k, an_j \in fl_i, an_k \in fl_i, fl_i \in AL, an_j$ *and* an_k *are in conflict*$\}$.

FL represents the set fact levels, AL represents the set action levels, E represents the set of edges, and ME is the set of mutual exclusions depicting the conflicts between the action nodes that instantiate mechanisms in $\mathcal{M}$. In the reminder if this chapter, we use the terms *action nodes* to denote vertices belonging to action levels and *fact nodes* to denote vertices belonging to fact levels.

Definition 37. ***(Correct Parallel Plan).*** *Given a planning graph* $\mathcal{PG}$ *of length* n*, a set of fact nodes* N *belonging to the fact level* fl_m *where* $m \leq n$*. A parallel plan* $\mathcal{P} = \{p_1, \cdots, p_{m-1}\}$ *correctly provides the set of fact nodes* N *if and only if the following conditions hold:*

1. $\forall p_i \in \mathcal{P} : p_i \subseteq al_i$.
2. $\forall p_i \in \mathcal{P}, \forall an_j \in p_i, \forall an_k \in p_i : an_j$ *and* an_k *are not conflicting mechanisms instances (Definition 28)* .
3. $\forall i \in [2, m-1], \forall an \in p_i : \Delta_{an} \subseteq \bigcup_{an_k \in p_{i-1}} \Sigma_{an_k}$.
4. $\forall i \in [2, m-1], \forall an \in p_i, \nexists p'_{i-1} \subset p_{i-1} : \Delta_{an} \subseteq \bigcup_{an' \in p'_{i-1}} \Sigma_{an'}$.
5. $N \subseteq \bigcup_{an_k \in p_{m-1}} \Sigma_{an_k}$.
6. $\nexists p'_{m-1} \subset p_{m-1} : N \subseteq \bigcup_{an \in p'_{m-1}} \Sigma_{an}$.

Condition (1) states that each p_i should be composed of action nodes belonging to the action level al_i of $\mathcal{PG}$. Condition (3) imposes that each action node in each p_i requires the effects of one or many action nodes in p_{i-1}. Using (4), we state that all action nodes belonging to p_{i-1} are needed to allow the application of the action nodes in p_i. Condition (5) states that the set of nodes N should belong to the effects of the action nodes in p_{m-1}. Finally, using (6) we state that all the action nodes in p_{m-1} are needed to provide N.

Lemma 1. *Given a planning graph $\mathcal{PG}$ of lenght n, a set of fact nodes $N = \{fn_1, \cdots, fn_r\}$ belonging to the fact level fl_m where $m \leq n$. Consider $\mathcal{PL}^N = \{\mathcal{P}_1^N = \{p_1^{1,N}, \cdots, p_{m-1}^{1,N}\}, \cdots, \mathcal{P}_q^N = \{p_1^{q,N}, \cdots, p_{m-1}^{q,N}\}\}$ being the set of parallel plans that correctly provide N, and $\mathcal{PL}^i = \{\mathcal{P}_1^i = \{p_1^{1,i}, \cdots, p_{m-1}^{1,i}\}, \cdots, \mathcal{P}_{q_i}^i = \{p_1^{q_i,i}, \cdots, p_{m-1}^{q_i,i}\}\}$ being the set of parallel plans that correctly provide fn_i $(1 \leq i \leq r)$. The following conditions hold:*

1. $\forall l \in [1,r], \forall i_l \in [1,q_l], \forall j \in [1,m-1], \exists k \in [1,q] : (\bigcup_{l=1}^{r} p_j^{i_l,l}) = p_j^{k,N}$
2. $\forall k \in [1,q], \forall j \in [1,m-1], \forall l \in [1,r], \exists i_l \in [1,q_l] : p_j^{k,N} = (\bigcup_{l=1}^{r} p_j^{i_l,l})$

The previous lemma proves the relations between the sets of parallel plans that can correctly provide the set of fact nodes N and the set of parallel plans that can correctly provide each fact node in N. Please refer to Section C.1 (Appendix C) for a proof of the previous lemma.

Definition 38. ***(Dominance relation).*** *A fact node fn_1 belonging to the fact level fl_1 dominates a fact node fn_2 (denoted dominates(fn_1,fn_2)) belonging to the fact level fl_2 iff for all $\mathcal{P}_i$ in the set of parallel plans $\mathcal{P}_1, \cdots, \mathcal{P}_n$ that correctly provides fn_2: $\exists p \in \mathcal{P}_i, \exists an \in p : fn_1 \in \Delta_{an}$.*

Informally speaking, the previous definition states that a fact node fn_1 dominates another fact node fn_2 if and only if all parallel plans that provide fn_2 need to use fn_1 to be considered as correct plans (Definition 37). In other words, if we remove the node fn_1 from the used planning graph, no parallel plan could correctly provide fn_2.

Lemma 2. *The dominance relation is transitive, i.e., if fn_1 dominates fn_2 and fn_2 dominates fn_3, then fn_1 dominates fn_3.*

We proved the previous lemma in Section C.2 (Appendix C).

Definition 39. ***(Security constraint).*** *A security constraint is a formula $C = \neg \bigwedge_{i=1}^{n} f_i$, where f_i are finite sets of ground literals.*

7.3 Planning Graph Tainting

Tainting is traditionally used in marking pieces of information to monitor how they are disseminated in a program or a system. They have been widely used to analyze how applications access sensitive data and how they process it. In our approach, tainting will be used in tracking security constraints violations in the planning graph. To meet this goal, each node in planning graph will be tainted using one or many taints. During the construction of the planning graph, each node (fact node/action node) in the planning graph can have one of the following form:

- **Untainted:** The node is created but not yet tainted;
- **Tainted:** the node is tainted using a set of taints (according to Definitions 42 and 44).

Definition 40. ***(Ground taint).*** *A ground taint t is an atomic taint such that $\nexists t_1, t_2 : (t_1 \wedge t_2 = t) \vee (t_1 \vee t_2 = t)$.*

Definition 41. ***(Safe node).*** *Given a set of constraints $\mathcal{C} = \{C_1, \cdots, C_n\}$, and a fact node fn belonging to the fact level fl of a planning graph. fn is a safe fact node regarding $\mathcal{C}$ (denoted safe_node(fn,C)) iff:*

$$\forall C \in \mathcal{C}, \nexists N \subset fl : \bigwedge_{fn_i \in N} fn_i \wedge fn \rightarrow \neg C$$

Definition 42. ***(Safe node tainting)*** *Consider a set of security constraints $\mathcal{C} = \{C_1, \cdots, C_l\}$, and a planning graph $\mathcal{G}$. Each fact node fn in $\mathcal{G}$ that does not violate any constraint in $\mathcal{C}$ and does not belong to any combination of nodes that violates any constraint $\mathcal{C}$ is tainted using $T_{fn} = \{t_1^{fn}, \cdots, t_l^{fn}\}$ where $\forall i \in [1, l] : t_i^{fn} = \emptyset$.*

Definition 43. ***(Unsafe node)*** *Given a set of constraints $\mathcal{C} = \{C_1, \cdots, C_n\}$, and a fact node fn belonging to the fact level fl of a planning graph. fn is an unsafe fact node regarding a constraint $C \in \mathcal{C}$ (denoted unsafe_node(fn,C)) iff:*

$$\exists N \subset fl : \bigwedge_{fn_i \in N} fn_i \wedge fn \rightarrow \neg C$$

Definition 44. ***(Unsafe node tainting)*** *Consider a set of security constraint $\mathcal{C} = \{C_1, \cdots, C_l\}$, and a set of sets of fact nodes $\mathcal{N} = \{N_1, \cdots, N_n\}$ belonging to a fact layer in a planning graph. Each $N_i \in \mathcal{N}$ is composed of a finite set of fact nodes $\{fn_1^i, \cdots, fn_{m_i}^i\}$ that violates a security constraint $C_i \in \mathcal{C}$ ($\bigwedge_{j=1}^{m_i} fn_j^i \rightarrow \neg C_i$). Each node fn in $\mathcal{N}$ will be tainted with the set of taints $T = \{t_1^{fn}, \cdots, t_l^{fn}\}$ where the following conditions hold:*

1. $\forall t_i \in T, \quad if \quad fn \notin N_i$ *then* $t_i = \emptyset$,

2. $\forall t_i \in T, \quad if \quad fn \in N_i$ *then* t_i *where:*

   (a) t_i *is a ground taint,*

   (b) t_i *is unique, e.g., no unsafe fact node in the planning graph could have the same ground taint.*

   (c) $t_i \neq \emptyset$,

   (d) $\bigwedge_{fn \in N_i} t_i^{fn} \leftrightarrow t_{\neg C_i}$, *where* $t_{\neg C_i}$ *is a unique taint denoting the violation of* C_i *(denoted* $vio\_taint(t_{\neg C_i}, C_i)$*),*

   (e) $\forall N_i \in \mathcal{N}, \nexists N_i' \subset N_i : \bigwedge_{fn \in N_i'} t_i^{fn} \models \neg C$.

7.3.1 Taint Propagation Rules

One of the key factors in defining our best-effort planning approach is the definition of a propagation policy for the taint marks. Our taint propagation policy treats the taint marks of a planning graph nodes associated with security constraints violation. We describe how the policy works in the two following rules.

Rule 1:

Given a planning graph $\mathcal{PG}$ and an action node an belonging to an action level in $\mathcal{PG}$. Suppose that an is linked to n fact nodes $fn_1, \cdots, fn_n$ representing the precondition of an, ech fn_i is tainted with the set of taints $T_i = \{t_1^i, \cdots, t_l^i\}$. Then the set of taints of an is: $T_{an} = \{\bigoplus_{i=1}^{n} t_1^i, \cdots, \bigoplus_{i=1}^{n} t_l^i\}$ where $\bigoplus$ is defined as follows:

$$t_j \bigoplus t_k = \left\{ \begin{array}{lll} t_j & if & t_k = \emptyset \\ t_k & if & t_j = \emptyset \\ t_j \wedge t_k & & otherwise \end{array} \right\}$$

The previous rule describes how taints are propagated from a fact-level to an action-level in the planning graph. More precisely, between the fact nodes representing the preconditions of an instance of a mechanism and the action node representing the instance of the mechanism.

Rule 2:

Given a planning graph $\mathcal{PG}$ and a fact node fn belonging to a fact level fl_m in $\mathcal{PG}$ and having the set of taints $T_{fn} = \{t_1^{fn}, \cdots, t_l^{fn}\}$ (produced by a safe/unsafe node tainting). Suppose that fn is linked to n action nodes $an_1, \cdots, an_n$ ($\forall i \in [1,n] : fn \in \Sigma_{an_i}$) belonging to the action level al_{m-1}, and that each action node an_i is tainted with $T_i = \{t_1^i, \cdots, t_l^i\}$. Then $T_{fn} = \{t_1^{fn} \bigoplus (\bigotimes_{i=1}^{n} t_1^i), \cdots, t_l^{fn} \bigoplus (\bigotimes_{i=1}^{n} t_l^i)\}$ where $\bigotimes$ is defined as following:

$$t_i \bigotimes t_j = \left\{ \begin{array}{ll} \emptyset & if \quad t_i = \emptyset \ or \ t_j = \emptyset \\ t_j \vee t_j & otherwise \end{array} \right\}$$

The Rule 2 specifies the propagation of taints from an action level to a fact level in the planning graph. More exactly, it states how the taints are propagated from an action node representing an instance of a mechanism to a set of fact nodes representing its effects.

Algorithm 9 describes how taints are propagated in the different fact levels and action levels of the planning graph. For each fact level fl in the planning graph $\mathcal{G}$, we start by checking if it is the first in $\mathcal{G}$ (line 3). If not, obviously, there exists an action level al which contains actions producing the fact nodes in fl. The **foreach** loop (lines 4 to 12) describes the taint propagation from the action nodes in al to the fact nodes in fl using the propagation rule 2. The function ***in_edges*** returns the set of action nodes having *fact_node* as effect.

The **foreach** loop (lines 14 to 19) describes the security constraints violation checking and the tainting of the fact nodes that cause the violation of a security constraint (according to the Definition 44). In the line 20, we use the function ***taint_safe*** to taint all untainted fact nodes in fl with $''\emptyset''$ taints as we are sure that these fact nodes don't lead to violate any security constraint in $\mathcal{SC}$. In the last part of the Algorithm 9 (lines 21 to 32), we first check if fl is the last fact level in $\mathcal{G}$. If not, we can be sure that there exists an action level al containing action nodes that need a subset (or all) the fact nodes in fl to be performed. We use then the propagation rule 1 to propagate the taints from the fact nodes in fl to the action nodes in al.

Theorem 11. ***(Taint propagation complexity)*** *Given a planning graph $\mathcal{G}$ composed of n fact levels $fl_1, \ldots, fl_n$ and $n-1$ action levels $an_1, \cdots, an_{n-1}$, a set of m security constraints $\mathcal{SC}$ that should enforced. Suppose that each fact level fl_i in $\mathcal{G}$ is composed of p_i fact nodes , each action level an_i in $\mathcal{G}$ is composed of q_i action nodes The computational complexity of the Algorithm 9 is $O(n \times m \times p \times q)$, where $\forall i \in [1,n], j \in [1,n-1] : p_i \leq p,\ q_i \leq q,\ p \in \{p_1, \cdots, p_n\},$ and $q \in \{p_1, \cdots, p_{n-1}\}$.*

```
input : 𝒢   /* A planning graph (Graphplan) */
        𝒮𝒞 /* A set of security constraints*/
output: 𝒢_t  /* A tainted planning graph */
1  Main
2  foreach fl in 𝒢 do
3      if (fl ≠ first_fact_level(𝒢)) then
4          foreach fn in fl do
5              for i = 1 to |𝒮𝒞| do
6                  t_temp = ∅
7                  foreach an in in_edges(fn) do
8                      t_temp = t_temp ⊗ t_i^an
9                  endfch
10                 t_i^fn = t_i^fn ⊕ t_temp
11             end
12         endfch
13     end
14     foreach C in 𝒮𝒞 do
15         causes_nodes = get_causes_violation_nodes(fl,C)
16         if (causes_nodes ≠ ∅) then
17             taint_violation(causes_nodes,C)     /* Definition 44 */
18         end
19     endfch
20     taint_safe(fl \ causes_nodes)     /* Definition 42 */
21     if (fl ≠ last_fact_level(𝒢)) then
22         foreach fn in fl do
23             foreach an in outer_edges(fn) do
24                 if (T_an = untainted) then
25                     T_an = T_fn, continue
26                 end
27                 for i = 1 to |𝒮𝒞| do
28                     t_i^an = t_i^an ⊕ t_i^fn
29                 end
30             endfch
31         endfch
32     end
33 endfch
```

Algorithm 9: Taint propagation in the planning graph

The previous theorem is proved in Section C.3 (Appendix C).

Now we are ready to present our main theorem which prove that by checking the taints of a set of fact nodes N representing the set of goals to attend, we will be able to decide without analyzing the planning graph if there exists a parallel plan that correctly provides N without violating the security constraint.

Theorem 12. *Given a planning graph $\mathcal{G}$, a set of security constraints $\mathcal{C} = \{C_1, \cdots, C_l\}$, a set fact nodes $N = \{fn_1, \cdots, fn_n\}$ belonging to the fact level fl_m of $\mathcal{G}$ and tainted respectively with $T_1, \cdots, T_n$, and a set of all parallel plans $\mathcal{PL}^N = \{\mathcal{P}_1^N, \cdots, \mathcal{P}_r^N\}$ that correctly provide N. The following conditions hold:*

1. *If $\forall i \in [1, l], \exists t : \left((\bigwedge_{j=1}^{n} t_i^{fn_j}) \models t \right) \wedge$ vio_taint(t, C_i), then no parallel plan in $\mathcal{PL}$ could correctly provide $fn_1, \cdots, fn_n$ without violating C_i ;*
2. *If $\forall i \in [1, l], \nexists t : \left((\bigwedge_{j=1}^{n} t_i^{fn_j}) \models t \right) \wedge$ vio_taint(t, C_i), then there exists at least one parallel plan in $\mathcal{PL}$ that correctly provides $fn_1, \cdots, fn_n$ without violating C_i.*

In the previous theorem, condition (1) states that if there exists a taint t which: (i) is formally satisfied by the conjunction of the taints of the set of fact nodes N, and (ii) indicates a violation of a security constraint C_i, we can be sure that no parallel plan in $\mathcal{PL}$ could correctly provide $fn_1, \cdots, fn_n$ without violating C_i. Condition (2) states that if there is no taint t which: (i) is formally satisfied by the conjunction of the taints of the set of fact nodes N, and (ii) indicates a violation of a security constraint C_i, then there exists at least one parallel plan in $\mathcal{PL}$ that correctly provides $fn_1, \cdots, fn_n$ without violating C_i. Please refer to Section C.4 (Appendix C) for the proof of the previous theorem.

7.4 Finding the Best Compromise

In the previous section, we proved that the application of our tainting approach over a planning graph allows us to decide whether a set of fact nodes representing a set of goals can be provided by a parallel plan without violating security constraints, and this without analyzing all the planning graph. We use this result to find the best trade-off between satisfying specified goals and ensuring security requirements. When we have to decide between maintaining security constraints or satisfying goals. It can be useful to assign some priority to them. For the sake of simplicity, we will use a simple quantitative approach by associated to each security constraint a numeric weight representing the cost of its violation and to each goal a numeric weight representing the benefit of its satisfaction.

Definition 45. ***(Best compromise).*** *Given a set goals $\mathcal{G} = \{G_1, \cdots, G_n\}$ that should be satisfied, a set of security constraints $\mathcal{C} = \{C_1, \cdots, C_m\}$ that should enforced, and a set of sets of fact nodes $N_1, \cdots, N_n$ that respectively satisfy $G_1, \cdots, G_n$. Suppose that a weight w_i^+ is assigned to each goal G_i and that a weight w_j^- is assigned to each constraint C_j ($1 \leq i \leq n,\ 1 \leq j \leq m,\ w_i^+ \geq 0, w_j^- \geq 0$). The best compromise between satisfying $\mathcal{G}$ and ensuring $\mathcal{C}$ is to find a plan that satisfy $\mathcal{G}'$ and ensure $\mathcal{C}'$ where the following conditions hold:*

1. $\mathcal{G}' \subseteq \mathcal{G}$ *and* $\mathcal{C}' \subseteq \mathcal{C}$*;*
2. $Score(\mathcal{G}', \mathcal{C}') = \sum_{G_i \in \mathcal{G}'} w_i^+ - \sum_{C_i \in \mathcal{C} \backslash \mathcal{C}'} w_i^-$, *where* $\forall C_i \in \mathcal{C}\backslash\mathcal{C}', \exists t :$ $\Big((\bigwedge_{G_i \in \mathcal{G}'} \bigwedge_{fn \in N_i} t_i^{fn}) \models t \Big) \wedge vio\_taint(t, C_i)$*;*
3. $\nexists \mathcal{G}'', \mathcal{C}'' : \mathcal{G}'' \subseteq \mathcal{G} \ \wedge\ \mathcal{C}'' \subseteq \mathcal{C} \ \wedge\ Score(\mathcal{G}'', \mathcal{C}'') > Score(\mathcal{G}', \mathcal{C}')$.

Condition (1) states that $\mathcal{G}'$ is a subset of $\mathcal{G}$ and $\mathcal{C}'$ is a subset of $\mathcal{C}$. Condition (2) states the function we use to compute the score of the satisfaction compromise represented by $(\mathcal{G}', \mathcal{C}')$. It is the difference between the sum of the weight of each goal in $\mathcal{G}'$ ($\mathcal{G}'$ is the set of satisfied goals) and the sum of the weight of each violated security constraint ($\mathcal{C}\backslash\mathcal{C}'$ represents the set of violated constraints). Condition (3) makes sure that no other compromise $(\mathcal{G}'', \mathcal{C}'')$ can provide a score greater than the one provided by $(\mathcal{G}', \mathcal{C}')$.

The problem of finding the best compromise between satisfying specified goals and ensuring security constraints is NP-hard. This is formally stated by the following theorem.

Theorem 13. *The problem of finding the best compromise between satisfying a set of goals and ensuring a set security constraints is NP-hard.*

The previous theorem is proved in Section C.5 (Appendix C).

Since the problem of finding the best compromise between satisfying a set of goals and ensuring a set security constraints is NP-hard, we use an heuristic resolution strategy to solve such a problem with a reasonable computational effort.

7.4.1 Heuristic Search

We use an heuristic in order to find a near-optimal compromise between satisfying a set of goals and ensuring a set of security constraints. It is based on a constructive

method consisting of building a solution to the problem step by step from scratch. The constructive method to be used is based on choosing for each iteration, the best goal to satisfy.

```
   input :
           G = {G_1, ··· , G_n}     /* A set of goals */
           C = {C_1, ··· , C_m} /* A set of security constraints*/
   output:
           (G_b, C_b)     /* A near-optimal compromise */
 1 Main
 2 G_b = ∅ ; C_b = ∅
 3 while true do
 4     G_b = NULL;  C_{G_b} = ∅
 5     foreach G in G do
 6         C_temp = get_enforced_Constraint(G_b ∪ G)
 7         if G_b = NULL then
 8             if Score(G_b ∪ G, C_temp) > Score(G_b, C_b) then
 9                 G_b = G
10                 C_b = C_temp
11             end
12             continue
13         end
14         if Score(G_b ∪ G, C_temp) > Score(G_b ∪ G_b, C_{G_b}) then
15             G_b = G
16             C_b = C_temp
17         end
18     endfch
19     if G_b = NULL then
20         break
21     end
22     G_b = G_b ∪ G_b
23     C_b = C_{G_b}
24     G = G\G_b
25 end
```

Algorithm 10: Finding the near optimal compromise

Definition 46. ***(i-th best goal).*** *Given a set goals $\mathcal{G} = \{G_1, \cdots, G_n\}$ that should be satisfied, a set of security constraints $\mathcal{C} = \{C_1, \cdots, C_m\}$ that should enforced. $G_b^i \in \mathcal{G}$ is the i-th best goal to satisfy iff the following conditions hold:*

1. *$G_b^i \notin \mathcal{G}_b$, where $\mathcal{G}_b = \{G_b^j \mid 1 \leq j \leq i-1\}$;*

2. $Score(\mathcal{G}_b \cup G_b^i, \mathcal{C}_i) \geq Score(\mathcal{G}_b, \mathcal{C}_{i-1})$, *where* $\mathcal{C}_i$ *represents the set of ensured constraints in the i-th iteration ;*

3. $\nexists G_j \in \mathcal{G} : G_j \neq G_b^i \ \wedge \ G_j \notin \mathcal{G}_b \ \wedge \ Score(\mathcal{G}_b \cup G_j, \mathcal{C}_j) \geq Score(\mathcal{G}_b \cup G_b^i, \mathcal{C}_i)$

The previous definition describes the best goal to satisfy in each iteration of our constructive method by the one satisfying two conditions. Condition (1) states that the best goal to satisfy in the $i-th$ iteration should not belong to the set of goals chosen in the previous $i-1$ iterations. Condition (2) states that the score representing the compromise between satisfying the set of goals in $(\mathcal{G}_b \cup G_b^i)$ and ensuring the set of security constraints $\mathcal{C}_i$ should be greater of equal than the score representing the best compromise chosen in the $(i-1)th$ iteration. Condition (3) makes sure that no other goal in $\mathcal{G} \backslash \mathcal{G}_b$, when satisfied, can provide a better compromise between satisfying goals in $\mathcal{G}$ and ensuring security constraints in $\mathcal{C}$, than the compromise provided by satisfying G_b^i.

Algorithm 10 illustrates how to use our heuristic constructive method to get the best set of goals $\mathcal{G}_b$ to satisfy and the best set of security constraints $\mathcal{C}_b$ to satisfy.

Once we get this result, the idea is to remove from the planning graph all the fact-nodes nodes that violates a constraint in $\mathcal{C}_b$, then we can use mechanisms planning approach presented in Chapter 6 to search the plan of mechanisms that satisfies the best compromise we got using our heuristic constructive method.

7.5 Conclusion

In this chapter, we present an approach combining graph-based planning techniques with a data tainting-based technique to find a best-effort solution for security mechanisms planning under security policy. We proved that our tainting technique can be used to track security requirements violation over a planning graph allowing us to get the best tradeoff between maintaining security requirements and satisfying intended goals, and this by analyzing only the propagated taints in the planning graph. Our future work will include the implementation of our approach.

CHAPTER 8

Conclusions and Perspectives

In conclusion, we give an overview on how the different research objectives presented in the introduction have been followed as well as the different contributions which have resulted. Afterwards, we reflect on how our contributions can be improved and provide new research directions.

Our main objective in this thesis was to propose new approaches ensuring data security in cloud environments. We build upon the fact that outsourced data owners do not fully trust cloud service providers.

The first objective of this thesis consists of defining new methods to improve data confidentiality in cloud environments, while allowing an efficient processing of the outsourced data. To meet this objective, we proposed two different approaches.

In Chapter 3, we introduce our first contribution [Bkakria et al. 2013a]. In [Bkakria et al. 2013a], we propose an approach combining data fragmentation and encryption to protect the confidentiality of outsourced multi-relational databases. It improves existing approaches [Ciriani et al. 2007, Ciriani et al. 2009] assuming a strong and seldom satisfied in real environments assumption, saying that the data to be outsourced is represented within a single relation schema. The same contribution also permits outsourced data owners to process fragmented databases through the definition of a secure and effective technique for querying the data distributed on several service providers. Finally, in [Bkakria et al. 2013b] we improve the security of the querying technique in order to protect data confidentiality under a collaborative Cloud service providers model.

Second, we present in Chapter 4 a policy-based configuration framework [Bkakria et al. 2014b] that allows a data owner to specify the set of security and utility requirements over the data to be outsourced. We then provide an efficient method permitting to detect conflicts between confidentiality requirements (e.g., the set of sen-

sitive information) and utility requirements (e.g., SQL queries that should be executed over the encrypted data) specified in the policy to be applied over the outsourced data. We define the best combination of encryption schemes that can satisfy a specified policy and proved that the problem of finding such a combination is NP-hard. Finally, we propose an heuristic polynomial-time algorithm for finding a combination of encryption schemes that satisfies a near optimal trade-off between confidentiality requirements and utility requirements.

The second objective of this thesis is built upon two main reasons: First, security and utility requirements that might be specified by data owners are in most of cases heterogeneous (e.g., confidentiality requirements, privacy requirements, ownership requirements, etc.). Second, Many security mechanisms allowing to enforce those requirements have been defined. The challenge then is to figure out the combination of security mechanisms that should be used. The objective consists in designing support tools that allow data owners to easily specify their security and utility requirements and automatically choose the best set of security mechanisms, and the best way to combine them (e.g. the best order in which they are applied) to get the best trade-off between complexity, security and utility in the final choices. To this end, we proposed three contributions.

We define a formal model relying on an expressive language allowing to: (1) formally specify a system composed of involved entities (e.g., data owner, Cloud server administrator, external adversary, etc.) and the data structure on which the policy should be enforced; (2) formally express as finely as possible the policy defined by the data owner; And (3), formally express existing security mechanisms that can be used to fulfill the requirements that might be requested by data owners.

As a second step, in Chapter 6 we define a reasoning method for the formal model we previously design allowing outsourced data owner to automatically figure out the combination of security mechanisms providing the near optimal trade-off between the security and the utility of the data to be outsourced and the complexity of the application of the chosen combination over the used system. Then, we implemented a proof of concept of our reasoning method to demonstrate the feasibility of our proposal and gave support to our given theoretical complexity measurements.

Out last contribution was presented in Chapter 7. It extends the reasoning method proposed in Chapter 6 by overcoming the all or nothing satisfaction property [1] of our reasoning method. It proposes an approach that associates data tainting method with graph planning analysis to get the mechanisms execution plan that provides the best

[1] In our reasoning method, security and utility requirements are either wholly satisfied or wholly violated, which allows our reasoning method only to deal with limited-scale policies

compromise between security constraints to enforce and the set of goals to satisfy over the outsourced data.

Perspectives

The research described in this thesis can be extended along several directions.

- In our proposed formal model (Chapter 5), we focused on five kinds of security requirements: Confidentiality, privacy, integrity, traceability, and ownership. In fact, some other interesting security requirements, such as authentication of both data and entities, data freshness, and proof of possession, might be requested by outsourced data owners. One possible perspective is to extend our formal model to allow outsourced data owners to deal with those security requirements.
- The specification of the policy to be applied over the outsourced data, as well as its defined refinement method, relies on relations between the objects composing the data to be outsourced. One of the most prominent challenge is then to adopt our formal model to be able to deal with outsourced unstructured big data.
- Implementation and performance evaluation of our approach that associates data tainting method with graph planning analysis to get the mechanisms execution plan that provides the best compromise between security constraints to enforce and the set of goals to satisfy over the outsourced data (Chapter 7). It would be interesting to evaluate its performances regarding the number of data objects to be outsourced, the number of security mechanisms that can be used, the number of entities involved in the outsourcing scenario.

To conclude, I would like to say that this research has been a great opportunity to investigate a wide variety of concepts, models and technologies in the information security domains. We provided novel approaches in response to the outsourced data security challenges, and we have shown that our proposed work is encouraging research field.

Finally, we believe that outsourced data security is still plenty of challenges and of paramount importance, and several research problems stay to be figured out and investigated.

APPENDIX A

Expression et déploiement de politiques de sécurité intégrées pour données externalisées

A.1 Introduction

L'externalisation des données donne lieu à de nombreux problèmes de sécurité, principalement, en raison de la perte du contrôle physique sur les données externalisées. D'un côté, faire respecter la confidentialité des données dans les environnements de stockage Cloud devient plus difficile lorsque les données sont stockées et gérées par des tiers non fiables. Une solution possible consiste à chiffrer les données qui seront externalisées sur la machine du propriétaire de données (qui est censé être fiable) avant de télécharger ces données sur le serveur de stockage Cloud. Le chiffrement des données externalisées est considéré comme étant la dernière ligne de défense efficace pour protéger la confidentialité des données à la fois des utilisateurs externes non autorisés et les administrateurs malveillants des serveurs de stockage Cloud. De toute évidence, si les clés de chiffrement ne sont pas compromises par un pirate ou un administrateur malveillant qui gère le serveur Cloud, la confidentialité des données externalisées reste assurée. Cette solution est inutile lorsqu'il s'agit d'externaliser de grandes bases de données de production.

Le premier objectif de cette thèse est de définir de nouvelles solutions permettant d'assurer le meilleur compromis entre la confidentialité et l'utilité des données externalisées. Pour atteindre cet objectif, nous proposons les contributions suivantes :

- Une première approche [Bkakria et al. 2013a, Bkakria et al. 2013b] permettant la protection de la confidentialité des informations sensibles stockées dans des bases de données multirelationnelles. Notre approche améliore une approche existante [Ciriani et al. 2007, Ciriani et al. 2009] basée sur la combinaison des techniques de fragmentation des données et des techniques de chiffrement des données.

- Une deuxième approche [Bkakria et al. 2014b] permettant dans un premier temps à un propriétaire de données de spécifier la politique de sécurité à appliquer sur ses données qui seront externalisées. Dans un second temps, l'approche proposée choisit automatiquement l'ensemble des mécanismes de chiffrement qui assure le meilleur compromis entre la confidentialité et l'utilité des données externalisées.

Par la suite, en analysant quelques scénarios d'externalisation des données. Nous nous rendons compte que les exigences de sécurité et d'utilité spécifiées par les propriétaires de données sont différentes dans chaque scénario. En outre, ces exigences de sécurité sont dans certains cas hétérogènes (p. ex., des exigences de confidentialité, des exigences en matière de vie privée, des exigences relatives au droit d'auteur, etc.).

Le deuxième objectif de cette thèse consiste à concevoir une solution permettant aux propriétaires de données de définir des exigences de sécurité hétérogènes et choisir automatiquement le meilleur ensemble de mécanismes de sécurité, et la meilleure façon de les combiner (p. ex. le meilleur ordre dans lequel ils sont appliqués) pour obtenir le meilleur compromis entre la complexité, la sécurité et l'utilité des données dans le choix final. Pour atteindre cet objectif, nous proposons les contributions suivantes :

- Dans un premier temps, en utilisant la logique temporelle épistémique de premier ordre (LTL épistémique), nous définissons un modèle formel [Bkakria et al. 2014a] permettant de : (1) modéliser le système composé des entités impliquées dans le processus d'externalisation de données (p. ex., le propriétaire de données, le fournisseur de stockage Cloud, adversaire externe, etc.) et de la structure des données sur laquelle la politique de sécurité doit être appliquée. (2) Exprimer aussi finement que possible les exigences de sécurité et d'utilité définies par le propriétaire de données. Ensuite, nous définissons une méthode de raisonnement pour notre modèle formel permettant de déterminer la combinaison de mécanismes de sécurité qui déploie efficacement la politique de sécurité et d'utilité définie.

- Dans second temps, Nous définissons une approche qui améliore [Bkakria et al. 2014a] en prenant en considération les conflits qui peuvent survenir entre les mécanismes de sécurité. Cette approche utilise une méthode basée sur les graphes de planification afin de trouver la combinaison de mécan-

ismes de sécurité offrant un compromis optimal entre la sécurité et l'utilité des données externalisées et la complexité de son application sur le système utilisé.

- En utilisant notre méthode raisonnement proposée dans les précédentes contributions, les politiques de sécurité définies sur les données externalisées sont soit totalement satisfaites ou violées. Notre dernière contribution surmonte cette limitation par l'utilisation d'une méthode reposant sur le marquage des données afin d'obtenir le compromis optimal entre les buts à satisfaire (p. ex. externaliser les données) et les contraintes de sécurité à assurer.

A.2 Préserver la confidentialité des bases de données multirelationnelles en combinant la fragmentation et le chiffrement des données

Notre approche de combinaison de la fragmentation et du chiffrement pour protéger la confidentialité des bases de données composées de plusieurs relations est illustrée par la Figure A.1.

Dans notre approche, différemment à ce qui a été considéré dans [Ciriani et al. 2007, Ciriani et al. 2009], on considère un scénario dans lequel les données sont enregistrées dans plusieurs tables relationnelles. La politique de confidentialité à déployer sur ces données est spécifiée en utilisant trois types de contraintes de confidentialité.

Contrainte de type Singleton : Elle est représentée par un ensemble contenant un seul attribut, ce type de contrainte de confidentialité signifie que les valeurs de l'attribut en question sont sensibles et doivent être protégés.

Contrainte d'association : Ce type de contrainte de confidentialité est représenté par un sous-ensemble d'attributs. Sémantiquement, cette contrainte signifie que l'association des valeurs de ces attributs est sensible et doit être protégée.

Contrainte Inter-tables : Elle est représentée par un couple de relations appartenant à la base de données à externaliser. L'utilisation de ce type de contrainte assure la protection de l'association reliant les deux relations concernées par la contrainte.

Ces différents types de contraintes de confidentialité sont satisfaites via l'utilisation de la fragmentation et du chiffrement des données.

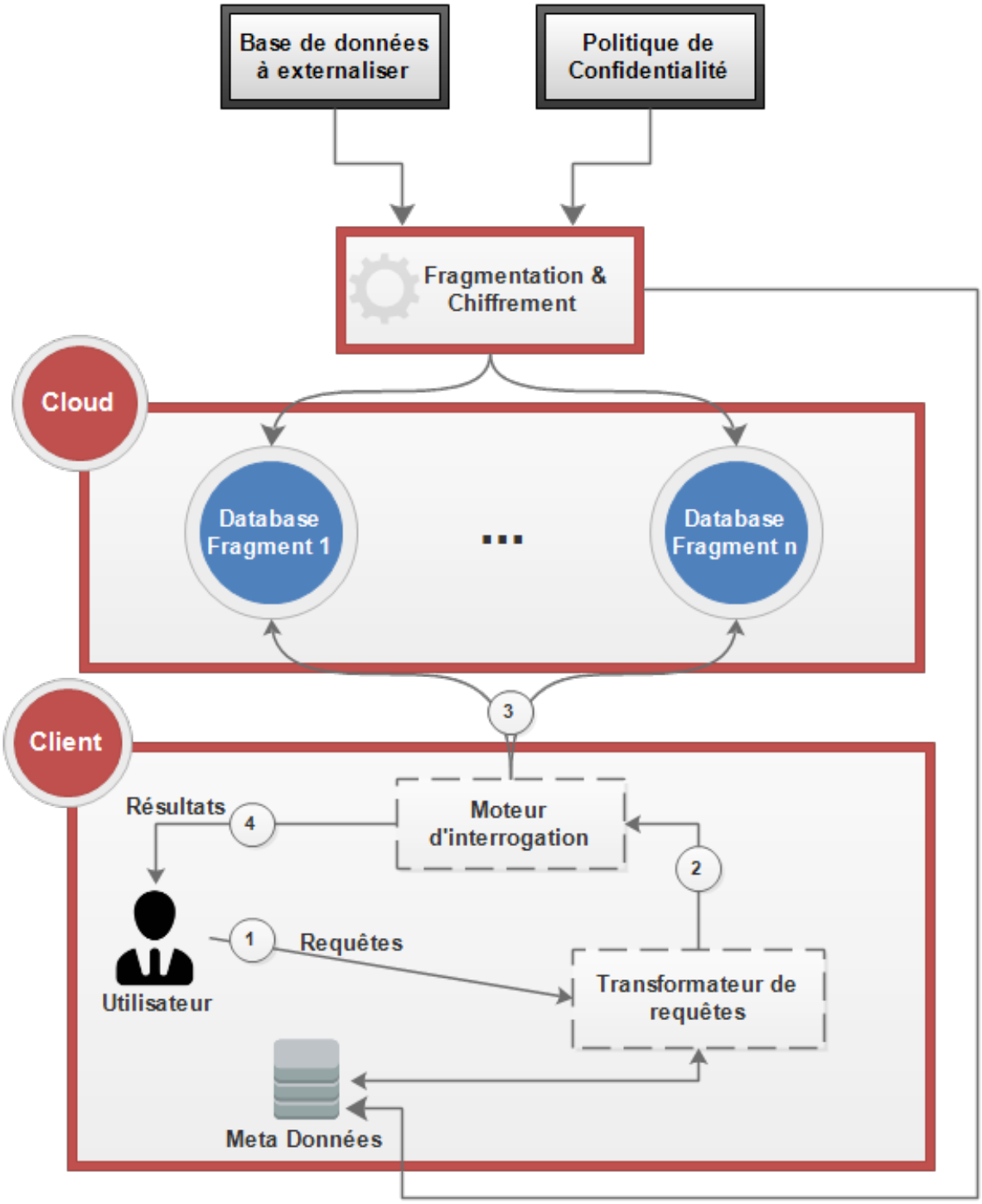


Figure A.1 – Architecture de fragmentation et d'interrogation de données

- Le chiffrement est appliqué au niveau des attributs. Un attribut est chiffré, signifie que toutes ses valeurs sont chiffrées.

- La fragmentation est aussi appliquée au niveau des attributs. Elle consiste à diviser un ensemble d'attributs pour garantir que ces attributs ne seront pas visibles ensemble dans le même fragment.

Pour permettre l'utilisation des données externalisées, nous avons défini une méthode sécurisée d'interrogation des données distribuées dans plusieurs serveurs Cloud. Afin d'exécuter une requête, l'utilisateur l'envoie au transformateur des requêtes. En se basant sur les métas-données qui contiennent la structure de la base de données initiale et les structures des fragments, le transformateur de requêtes analyse syntaxiquement la requête, puis construit un plan d'interrogation de données optimisées. Le moteur d'interrogation exécute le plan sur les différents fragments et envoie le résultat à l'utilisateur. Finalement, nous avons proposé une technique permettant d'associer, de façon efficace et sécurisée, les données des différents fragments. Elle repose sur l'utilisation d'un protocole de retrait d'informations privée par mot clé.

A.3 La combinaison des mécanismes de chiffrement pour assurer la confidentialité des données externalisées

Cette contribution définit une approche permettant au propriétaire de données, dans un premier temps, de spécifier les exigences de confidentialité et les fonctionnalités à assurer sur les données externalisées, puis dans un second temps, de sélectionner l'ensemble de mécanismes de chiffrements qui assure un compromis optimal entre la sécurité et l'utilité des données externalisées.

A.3.1 Spécification de la politique à déployer

Pour permettre au propriétaire de données de spécifier la politique à déployer sur les données externalisées, nous avons défini trois types de contraintes :

- Des contraintes de confidentialité permettant de spécifier les attributs sensibles à protéger;
- Des contraintes de niveau de confidentialité permettant de spécifier pour chaque attribut sensible un seuil minimum de confidentialité à assurer. Trois niveaux de confidentialité peuvent être utilisés pour classifier les données.

  - Le niveau “top secret” signifiant que toute fuite d’information au sujet des données va causer de graves dommages, on associe ce niveau au niveau RND qui représente le niveau de confidentialité assurée par un chiffrement probabiliste qui offre la sécurité sémantique des données.
  - Le niveau “secret” signifiant que certaines informations sur les valeurs de données peuvent être divulguées si elles ne conduisent pas à révéler les valeurs elles-mêmes, ce niveau est associé au niveau “DET” représentant le niveau de sécurité assurée par un chiffrement déterministe. Un chiffrement déterministe ne permet pas de protéger les données contre les attaques à clairs connues ainsi que les attaques fréquentielles. Par conséquent, nous supposons, dans notre approche, que le niveau de sécurité “DET” est moins sûr que le niveau “RND”.
  - Le niveau “confidentiel” est associé au niveau de confidentialité “OPE” représentant le niveau assuré par un chiffrement qui préserve l’ordre. Il a été conclu que les chiffrements qui préservent l’ordre permettent au moins la fuite de la moitié des bits du texte clair. Pour ces raisons, nous supposons dans notre approche que le niveau de sécurité “OPE” est moins sûr que le niveau “DET”.

- Des contraintes d’utilité permettant des spécifier les fonctionnalités à assurer sur les données externalisées. Dans la plupart des cas, les bases de données externalisées sont utilisées par des applications. Ces applications peuvent être analysées pour extraire l’ensemble des requêtes qui seront exécutées sur les données externalisées, or, à partir de ces requêtes, on peut facilement récupérer l’ensemble de fonctionnalité à assurer sur les données (p. ex., la recherche avec égalité “$=$”, la recherche par préservation d’ordre “$\leq, \geq$”, le calcul “AVG, SUM, $+$, $\times$”, etc.).

A.3.2 Déploiement de la politique

Afin de déployer une politique spécifiée, nous avons défini, dans un premier temps, une méthode permettant d’étudier la consistance de la politique. Il s’agit de chercher des conflits dans la politique à appliquer. Ces conflits surviennent lorsque les objectifs de deux ou plusieurs contraintes ne peuvent pas être satisfaits simultanément. Nous avons prouvé par la suite que trouver la combinaison de mécanismes de chiffrement qui assure le meilleur compromis entre la confidentialité et l’utilité des données externalisée est NP-difficile. Par conséquent, nous ne pouvons pas nous attendre à être en mesure de résoudre de façon optimale des instances de taille arbitraire du problème. Afin de pallier cette limitation, nous avons proposé une méthode heuristique basée

sur la construction de la solution étape par étape à partir de zéro. Pour chaque attribut sensible dans la base de données externalisée, notre méthode choisit le meilleur schéma de chiffrement qui satisfait les contraintes définies sur l'attribut en question. Ce meilleur schéma de chiffrement à deux caractéristique. Premièrement, il satisfait le seuil minimal de sécurité défini sur l'attribut sensible. Deuxièmement, il fournit le plus grand nombre de fonctionnalités comparé aux autres schémas de chiffrement. La complexité de l'algorithme implémentant notre méthode constructive est polynomiale.

A.4 Spécification et déploiement des politiques de sécurité hétérogènes sur les données externalisées

Le deuxième objectif de cette thèse est fondé sur deux raisons principales : d'abord, les exigences de sécurité et l'utilité qui peuvent être spécifiées par les propriétaires des données sont dans la plupart des cas hétérogènes (p. ex., des exigences de confidentialité, des exigences en matière de vie privée, des exigences de protection des droits d'auteurs, etc.). Deuxièmement, de nombreux mécanismes de sécurité permettant de faire respecter ces exigences ont été définis. Le défi consiste alors à déterminer la combinaison de mécanismes de sécurité qui doit être utilisée. Notre objectif dans cette partie consiste à concevoir une solution permettant aux propriétaires de données de définir aisément leurs exigences des sécurités et de fonctionnalité et choisir automatiquement le meilleur ensemble de mécanismes de sécurité, et la meilleure façon de les combiner (p. ex. le meilleur ordre dans lequel ils sont appliqués) pour obtenir le meilleur compromis entre la sécurité et l'utilité des données externalisées et la complexité de déploiement de la solution trouvée. La solution que nous avons proposée est illustrée par la Figure A.2.

Dans un premier temps, nous avons défini un modèle formel basé sur la logique temporelle épistémique de premier ordre. En effet cette logique est la composition de la logique de premier ordre, la logique temporelle, et la logique épistémique. Les raisons pour lesquelles on a combiné ces trois systèmes formels sont : premièrement, le besoin de la puissance d,Äôexpression de la logique de premier ordre pour pouvoir formellement spécifier le modèle d,Äôexternalisation des données qui comporte, la structure de données ou le système d,Äôinformation à externaliser. Deuxièmement, le besoin des opérateurs temporels fournis par la logique temporelle afin de pouvoir spécifier des contraintes de sécurité sur un ou plusieurs moments du processus d,Äôexternalisation de données. Finalement, lorsqu'on traite des problèmes de sécurité, la capacité à définir,

qui connaît quoi, devient particulièrement importante. La logique épistémique est donc utilisée pour spécifier les connaissances des entités impliquées dans le processus d,Äôexternalisation de données. Le modèle formel défini nous a permit de:

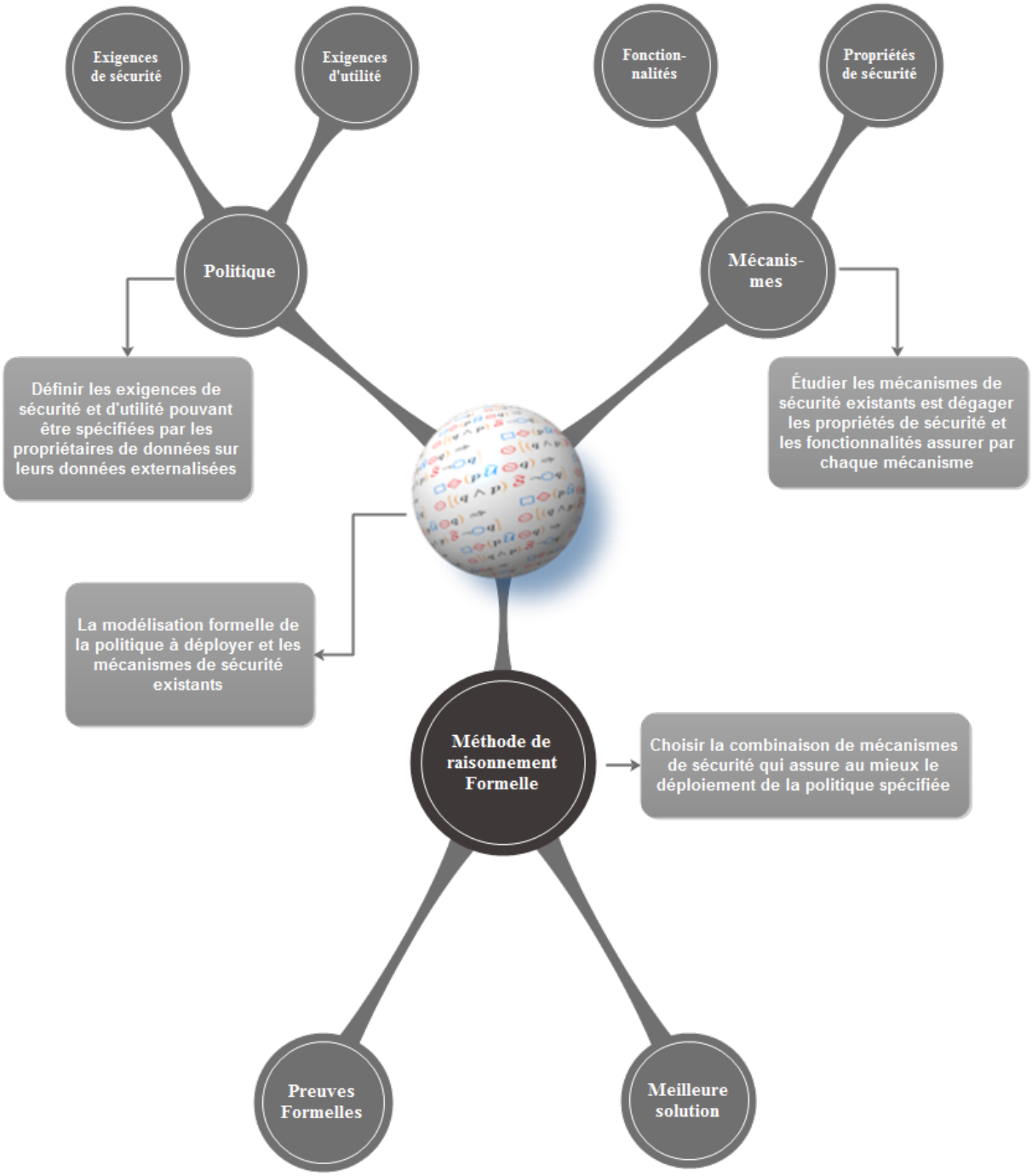


Figure A.2 – Architecture de fragmentation et d'interrogation de données

- Formellement, spécifier un système composé des différentes entités impliquées dans le processus d'externalisation de données (p. ex., propriétaire de données, admin-

istrateur de serveur Cloud, adversaire externe, etc.) et de la structure de données sur lequel la politique devrait être appliquée.

- Formellement, exprimer aussi finement que possible la politique définie par le propriétaire de données sur ses données externalisées.
- Formellement, exprimer les mécanismes de sécurité existants qui peuvent être utilisés pour satisfaire les exigences de sécurité et d'utilité qui pourraient être demandées par les propriétaires de données.

L'un des plus grands défis à relever dans cette deuxième partie de la thèse consiste à définir une méthode de raisonnement efficace pour le modèle formel que nous avons défini. Cette méthode a pour but de déterminer la combinaison de mécanismes de sécurité qui déploie la politique choisie sur les données qui seront externalisées. Pour atteindre cet objectif, dans un premier temps, nous avons défini une méthode de raisonnement composée de quatre étapes afin de récupérer l'ensemble des mécanismes de sécurité permettant d'assurer les exigences spécifiées du propriétaire de données. Puis, dans une deuxième étape, nous avons étendu une approche basée sur les graphes de planification pour construire un graphe représentant toutes les transformations possibles du système résultant de l'application de l'ensemble des mécanismes de sécurité obtenu antérieurement par notre méthode de raisonnement. Dans une troisième étape, nous avons défini une méthode permettant de chercher le plan d'exécution de mécanismes de sécurité quasi optimale permettant de transformer le système cible de son état initial (état dans lequel les données sont toujours stockées par leur propriétaire), à un état dans lequel les données sont externalisées et la politique définie par le propriétaire de données est déployée. Finalement, nous avons amélioré notre méthode de raisonnement déjà présenté en combinant l'utilisation des graphes de planification avec des techniques de marquage de données. L'intuition derrière le marquage des noeuds des graphes de planification est de pouvoir tracer les violations des contraintes de sécurité dans ces graphes. L'analyse des marques des noeuds des graphes de planification nous a permis de trouver le meilleur compromis entre satisfaire les objectifs de propriétaire de données (externaliser les données et assurer les fonctionnalités demandées sur les données) et assurer les exigences de sécurités spécifiées.

A.5 Conclusion

Notre principal objectif dans cette thèse est de proposer de nouvelles approches pour assurer la sécurité des données externalisées en se basant sur le fait que les propriétaires

de données hésitent à faire confiance aux fournisseurs de services pour la sécurisation de leurs données sensibles externalisées. Les recherches menées dans cette thèse peuvent être étendues dans plusieurs directions:

- Èxtendre notre modèle formel pour permettre de spécifier et déployer d,Äôautres exigences de sécurité (p. ex., L'authentification de données et d'entités, l'actualisation des données, la preuve de possession, etc.),
- Adapter notre solution pour pouvoir l'appliquer sur des données non structurées (p. ex., BigData),
- Adapter notre modèle pour pouvoir l'utiliser dans la génération automatique des protocoles de sécurité.

APPENDIX B

The Specification of the System Used to Evaluate our Approach Proposed in Chapter 6

The system we used to evaluate our security mechanisms planning approach (Chapter 6) is specified using STRIPS-like planning domains [Fikes and Nilsson 1971] in two files: a domain file and a problem file.

The domain file describes the language to be used to specify a target system by defining the different predicates that are to be used in the specification. It includes also the specification of: a set of mechanisms that can be used to enforce a security policy, a set of axioms that will be used to infer new facts and knowledge about the used domain, and a set of security constraints that should be enforce over the target system. Listing B.1 shows the content of the used domain file.

The problem file contains three main parts: A first part describing the (typed) objects that compose the target system. A second part describing the initial state of the target system. And the last part describes the set of goals that should be reached. Listing B.2 shows the content of the used problem file.

Listing B.1 – The domain file used in the evaluation of our security mechanisms planning approach (Chapter 6)

```
(define (domain poseidon)
  (:requirements :strips :equality :typing)
  (:types OBJ USER COBJ PROP WATERMARK)
  (:predicates  (belongs ?owner - USER ?obj - OBJ)
                (hom_encrypted ?obj - OBJ ?key - KEY ?objenc - OBJ)
                (det_encrypted ?obj - OBJ ?key - KEY ?objenc - OBJ)
```

```
                (ope_encrypted ?obj - OBJ ?key - KEY ?objenc - OBJ)
                (knows ?obj - OBJ ?own - USER)
                (knows ?obj - COBJ ?own - USER)
                (knows ?obj - KEY ?own - USER)
                (knows ?obj - WATERMARK ?own - USER)
                (outsourced ?obj - OBJ ?own - USER)
                (outsourced ?obj - COBJ ?own - USER)
                (enc_key ?obj - KEY)
                (trusted ?user - USER)
                (provides ?u - USER ?obj - COBJ ?p - PROP)
                (provides ?u - USER ?obj - OBJ ?p - PROP)
                (watermarked ?obj - OBJ ?key - OBJ ?w - WATERMARK ?wobj - COBJ)
                (sign_of ?u - USER ?w - WATERMARK)
                (used ?k - OBJ)
                (used ?w - WATERMARK)
                (empty ?o - COBJ)
                (rel_db ?o - OBJ)
                (rel_table ?o - OBJ)
                (fingerprint ?o - OBJ ?w - WATERMARK)
                (data_distortion ?o - OBJ)
  )


  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;; List of mechanisms
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


  (:ACTION RND_encryption
        :PARAMETERS
                 (?o1 - OBJ
                  ?k - KEY
                  ?objenc - COBJ)
        :PRECONDITION
                (and (enc_key ?k)(not(used ?k))(empty ?objenc))
        :EFFECT
                (and (RND_encrypted ?o1 ?k ?objenc) (not (empty ?objenc))
                (used ?k) (not (data_distortion ?objenc)))
  )

  (:ACTION encrypt_Hom
        :PARAMETERS
                 (?o1 - OBJ
                  ?k - KEY
                  ?objenc - COBJ)
        :PRECONDITION
                (and (enc_key ?k)(not(used ?k))(empty ?objenc))
        :EFFECT
                (and (hom_encrypted ?o1 ?k ?objenc) (not (empty ?objenc))
                (used ?k) (not (data_distortion ?objenc)))
  )


 (:ACTION encrypt_det
        :PARAMETERS
                 (?o1 - OBJ
                  ?k - KEY
                  ?objenc - COBJ)
```

```
        :PRECONDITION
                (and (enc_key ?k)(not(used ?k))(empty ?objenc))
        :EFFECT
                (and (det_encrypted ?o1 ?k ?objenc) (not (empty ?objenc)) (used ?k)
                (not (data_distortion ?objenc)))
)

(:ACTION encrypt_ope
        :PARAMETERS
                 (?o1 - OBJ
                  ?k - KEY
                  ?objenc - COBJ)
        :PRECONDITION
                (and (enc_key ?k)(not(used ?k))(empty ?objenc))
        :EFFECT
                (and (ope_encrypted ?o1 ?k ?objenc) (not (empty ?objenc)) (used ?k)
                (not (data_distortion ?objenc)))
)

(:ACTION watermark_OW
        :PARAMETERS
                 (?o - OBJ
                  ?k - KEY
                  ?u - USER
                  ?w - WATERMARK
                  ?objenc - COBJ)
        :PRECONDITION
                (and (enc_key ?k)(belongs ?u ?o)(not(used ?k))(empty ?objenc))
        :EFFECT
                (and (watermarked ?o ?k ?w ?objenc)(sign ?u ?w)(not (empty ?objenc))
                 (not (data_distortion ?objenc)))
)
(:ACTION watermark_OW_TP
        :PARAMETERS
                 (?o - OBJ
                  ?k - KEY
                  ?u - USER
                  ?w - WATERMARK
                  ?objenc - COBJ)
        :PRECONDITION
                (and (enc_key ?k)(belongs ?u ?o)(not(used ?k))(empty ?objenc))
        :EFFECT
                (and (watermarked ?o ?k ?w ?objenc)(sign ?u ?w)(fingerprint ?o ?w)
                (not (empty ?objenc))(not (data_distortion ?objenc)))
)

(:ACTION anonymization
        :PARAMETERS
                 (?o - OBJ
                  ?objanon - COBJ)
        :PRECONDITION
                (and (rel_table ?o)(empty ?objanon))
        :EFFECT
                (and (anonymized ?o ?objanon)(not (empty ?objanon))
                (data_distortion ?objanon))
)

(:ACTION signature
```

```
        :PARAMETERS
                (?o - OBJ
                 ?user - USER
                 ?objs - COBJ)
        :PRECONDITION
                (and (knows ?o ?user)(empty ?objs))
        :EFFECT
                (and (signed ?o ?user ?objs)(not (empty ?objs)))
    )

    (:ACTION outsource
        :PARAMETERS
                (?o)
        :PRECONDITION
                (not (empty ?o))
        :EFFECT
                (knows ?o cloudServer)
    )

    (:ACTION send
        :PARAMETERS
                (?k - KEY
                ?u - USER)
        :PRECONDITION

        :EFFECT
                (knows ?k ?u)
    )

    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ;; List of axioms
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    (:AXIOM axiom1
        :PARAMETERS
                (?o1- OBJ
                 ?o2 - COBJ
                 ?k - KEY
                 ?u - USER)
        :PRECONDITION
                (and (enc_key ?k) (hom_encrypted ?o1 ?k ?o2) (knows ?k ?u)
                (knows ?o2 ?u) (not (data_distortion ?o2)))
        :EFFECT
                (knows ?o1 ?u)
    )

    (:AXIOM axiom2
        :PARAMETERS
                (?o1- OBJ
                 ?o2 - COBJ
                 ?k - KEY
                 ?u - USER)
        :PRECONDITION
                (and (enc_key ?k) (det_encrypted ?o1 ?k ?o2) (knows ?k ?u)
                (knows ?o2 ?u) (not (data_distortion ?o2)))
        :EFFECT
                (knows ?o1 ?u)
    )
```

```
181
    (:AXIOM axiom3
          :PARAMETERS
184                (?o1- OBJ
                    ?o2 - COBJ
                    ?k - KEY
187                 ?u - USER)
          :PRECONDITION
                  (and (enc_key ?k) (ope_encrypted ?o1 ?k ?o2) (knows ?k ?u)
190               (knows ?o2 ?u) (not (data_distortion ?o2)))
          :EFFECT
                  (knows ?o1 ?u)
193 )

    (:AXIOM axiom4
196       :PARAMETERS
                   (?s ?u - USER
                    ?o)
199       :PRECONDITION
                  (and (has_access ?u ?s) (knows ?o ?s))
          :EFFECT
202               (knows ?o ?u)
    )

205 (:AXIOM axiom5
          :PARAMETERS
                   (?o1- OBJ
208                 ?o2 - COBJ
                    ?k - KEY
                    ?u - USER)
211       :PRECONDITION
                  (and(enc_key ?k) (hom_encrypted ?o1 ?k ?o2) (knows ?o2 ?u))
          :EFFECT
214               (and (provides ?u ?o1 avg) (provides ?u ?o1 sum)
                  (provides ?u ?o1 addition))
    )
217
    (:AXIOM axiom6
          :PARAMETERS
220                (?o1- OBJ
                    ?o2 - COBJ
                    ?k - KEY
223                 ?u - USER)
          :PRECONDITION
                  (and(enc_key ?k) (det_encrypted ?o1 ?k ?o2) (knows ?o2 ?u))
226       :EFFECT
                  (provides ?u ?o1 equality)
    )
229
    (:AXIOM axiom7
          :PARAMETERS
232                (?o1- OBJ
                    ?o2 - COBJ
                    ?k - KEY
235                 ?u - USER)
          :PRECONDITION
                  (and(enc_key ?k) (ope_encrypted ?o1 ?k ?o2) (knows ?o2 ?u))
238       :EFFECT
```

```
                (provides ?u ?o1 order)
    )
    (:AXIOM axiom8
        :PARAMETERS
                 (?o- OBJ
                  ?u - USER)
        :PRECONDITION
                 (knows ?o ?u)
        :EFFECT
                (and (provides ?u ?o equality)(provides ?u ?o avg)
                (provides ?u ?o addition)(provides ?u ?o sum)(provides ?u ?o order))
    )

    (:AXIOM axiom9
        :PARAMETERS
                 (?o - OBJ
                  ?objwat - COBJ
                  ?k - KEY
                  ?w - WATERMARK
                  ?u - USER)
        :PRECONDITION
                (and (enc_key ?k) (watermarked ?o ?k ?w ?objwat) (knows ?k ?u)
                (knows ?objwat ?u) (not (data_distortion ?objwat)))
        :EFFECT
                (and (knows ?w ?u) (knows ?o ?u))
    )

    (:AXIOM axiom10
        :PARAMETERS
                 (?o - OBJ
                  ?objwat - COBJ
                  ?k - KEY
                  ?w - WATERMARK
                  ?u - USER)
        :PRECONDITION
                (and (enc_key ?k) (watermarked ?o ?k ?w ?objwat) (knows ?k ?u)
                (knows ?objwat ?u) (data_distortion ?objwat))
        :EFFECT
                (knows ?w ?u)
    )

    (:AXIOM axiom11
        :PARAMETERS
                 (?o - OBJ
                  ?objwat - COBJ
                  ?k - KEY
                  ?owner - USER
                  ?w - WATERMARK
                  ?u - USER)
        :PRECONDITION
                (and (watermarked ?o ?k ?w ?objwat) (knows ?w ?u)
                (sign ?owner ?w))
        :EFFECT
                (provides ?u ?o ownership)
    )

    (:AXIOM axiom12
        :PARAMETERS
```

```
                    (?o - OBJ
                     ?objwat - COBJ
                     ?k - KEY
                     ?owner - USER
                     ?w - WATERMARK
                     ?u - USER)
        :PRECONDITION
                (and (watermarked ?o ?k ?w ?objwat) (knows ?w ?u)
                (fingerprint ?o ?w))
        :EFFECT
                (provides ?u ?o tamper_detection)
  )

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;; List of constraints
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  (:CONSTRAINT c1
        :formula
                (and (sensitive o6) (knows o6 ?u) (not (trusted ?u)))
  )

  (:CONSTRAINT c3
        :formula
                (and (sensitive o1) (knows o1 ?u) (not (trusted ?u)))
  )

  (:CONSTRAINT c4
        :formula
                (and (sensitive o2) (knows o2 ?u) (not (trusted ?u)))
  )

  (:CONSTRAINT c5
        :formula
                (and (sensitive o5) (knows o5 ?u) (not (trusted ?u)))
  )

  ...
)
```

Listing B.2 – The problem file used in experimentation

```
(define (problem poseidonProblem)
  (:domain poseidon)
  (:objects o1 o2 o3 o4 o5 o6 o7 o8 o9 o10 o11 o12 o13 o14 o15 o16 o17 o18 - OBJ
            k1 k2 k3 k4- KEY
            to1 to2 to3 to4- COBJ
            w1 w2 w3 - WATERMARK
            owner u1 u2 u3 u4 cloudServer - USER
            addition order equality like avg sum tamper_detection ownership - PROP)
  (:init
        (enc_key k1) (enc_key k2) (enc_key k3) (enc_key k4) (enc_key k5)
        (not (used k1)) (not (used k2)) (not (used k3)) (not (used k4)) (not (used k5))
        (not (used w1)) (not (used w2)) (not (used w3)) (not (used w4)) (not (used w5))
        (empty to1) (empty to2) (empty to3) (empty to4) (empty to5)
        (has_access u1 cloudServer) (has_access u2 cloudServer)
        (not (empty o1)) (not (empty o2)) (not (empty o3)) (not (empty o4))
```

```
        (not (empty o5)) (not (empty o10)) (not (empty o6)) (not (empty o7))
        (not (empty o8)) (not (empty o9))
        (belongs owner o1) (belongs owner o4)
        (not (encrypted o1)) (not (encrypted o2))
        (not (knows k1 cloudServer)) (not (knows k2 cloudServer))
        (not (knows k3 cloudServer)) (not (knows k4 cloudServer))
        (not (knows o1 cloudServer)) (not (knows o2 cloudServer))
        (not (trusted cloudServer)) (trusted u1) (trusted u2)
        (sensitive o1) (sensitive o2) (sensitive o3) (sensitive o4)
  )
  (:goal (and
                (provides cloudServer o3 order)
                (provides cloudServer o2 avg)
                (provides cloudServer o1 addition)
                (provides cloudServer o5 sum)
                (knows o1 u2)
                (knows o5 u3)
                (provides u2 o1 ownership)
                (provides cloudServer o8 order)
                (provides cloudServer o7 avg)
                (provides cloudServer o11 addition)
                (provides cloudServer o15 tamper_detection)
                (knows o15 u2)
                (knows o11 u3)
        ))
)
```

APPENDIX C

Proofs of Theorems and Lemmas of Chapter 7

C.1 Proof of Lemma 1

(i) Proof is by induction (recurrence). Let us start by $j = m-1$. By assumption, $p_{m-1}^{1,i}, \cdots, p_{m-1}^{q_i,i}$ represent respectively all the combinations of action nodes that provide fn_i. Then obviously, the set of all combinations of action nodes that provides N is $\{\bigcup_{l=1}^{r} p_{m-1}^{i_l,l} \mid l \in [1,r], i_l \in [1,q_l]\}$. Then we deduce that (i) is true for $j = m-1$. Now let us assume that (i) is true for j=2. Then we have:

$$\forall l \in [1,r], \forall i_l \in [1,q_l], \exists k \in [1,q] : (\bigcup_{l=1}^{r} p_2^{i_l,l}) = p_2^{k,N} \tag{C.1}$$

Suppose that each $p_2^{i_l,l}$ is composed of the set of nodes $an_1^{i_l,l}, \cdots, an_{s_{i_l}}^{i_l,l}$, and that each $\Delta_{an_t^{i_l,l}} = N_t^{i_l,l}$ $(1 \leq t \leq s_{i_l})$. Let us denote $p_1^{N_t^{i_l,l}}, \cdots, p_{v_t}^{N_t^{i_l,l}}$ all the combinations of each action nodes that provides $N_t^{i_l,l}$. Then we can deduce that:

$$p_1^{i_l,l} = \{\bigcup_{t=1}^{s_{i_l}} p_w^{N_t^{i_l,l}} \mid w \in [1,v_t]\} \tag{C.2}$$

Therefore,

$$\bigcup_{l=1}^{r} p_1^{i_l,l} = \{\bigcup_{l=1}^{r} \bigcup_{t=1}^{s_{i_l}} p_w^{N_t^{i_l,l}} \mid w \in [1,v_t]\} \tag{C.3}$$

We can deduce from (C.1) that the set of action nodes $A_2^{k,N}$ that compose $p_2^{k,N}$ is equal to $\bigcup_{l=1}^{r} \bigcup_{t=1}^{s_{i_l}} an_t^{i_l,l}$ which allows as to deduce that:

$$p_1^{k,N} = \{\bigcup_{l=1}^{r} \bigcup_{t=1}^{s_{i_l}} p_w^{N_t^{i_l,l}} \mid w \in [1,v_t]\} \tag{C.4}$$

Finally, using (C.3) and (C.4) we deduce that $p_1^{k,N} = \bigcup_{l=1}^{r} p_1^{i_l,l}$. (ii) can also be proved by recursion using the same method used to prove (i).

C.2 Proof of Lemma 2

Since fn_2 dominates fn_3, suppose that $\mathcal{P}_1, \cdots, \mathcal{P}_n$ are the set of parallel plans that correctly provide fn_3 we can deduce that:

$$\forall \mathcal{P} \in \{\mathcal{P}_1, \cdots, \mathcal{P}_n\}, \exists p \in \mathcal{P}, \exists an \in p : fn_2 \in \Delta_{an} \tag{C.5}$$

Also, since fn_1 dominates fn_2, we can deduce that:

$$\forall \mathcal{P} \in \{\mathcal{P}_1, \cdots, \mathcal{P}_n\}, \exists p \in \mathcal{P}, \exists an \in p : fn_1 \in \Delta_{an}s \tag{C.6}$$

Then from (C.6) we can deduce that fn_1 dominates fn_3.

C.3 Proof of Theorem 11

The application of the rule 2 represented in Algorithm 9 by the **foreach** loop (lines 4 to 12) has, in the worst case, computational complexity $O(m \times p_i \times q_{i-1})$ as we can suppose that each fact node in the fact level fl_i is an effect of each action node of the action level an_{i-1}. The tainting of the nodes causing the violation of security constraints is performed in Algorithm 9 through the **foreach** loop (lines 14 to 19) and has, in the worst case, computational complexity $O(m \times p_i)$ since we suppose that all fact nodes in the fact level fl_i can be used to violate all security constraints in $\mathcal{SC}$. The complexity of the application of the rule 2 represented in Algorithm 9 by the **foreach** loop (lines 21 to 32) is in the worst case $O(m \times p_i \times q_i)$ since we can suppose that each fact node in the fact level fl_i is a precondition of each action node of the action level an_i. The computational complexity of the Algorithm 9 is therefore $O(n \times m \times p \times q)$, where $\forall i \in [1, n], j \in [1, n-1] : p_i \leq p \text{ and } q_i \leq q,\ p \in \{p_1, \cdots, p_n\},\ \text{and } q \in \{q_1, \cdots, q_{n-1}\}$.

C.4 Proof of Theorem 12

To prove the Theorem 12, we will use the following Three Lemmas.

Lemma 3. *Given a set of constraints $\mathcal{C} = \{C_1, \cdots, C_m\}$, a tainted planning graph $\mathcal{PG}_t$, and a fact node fn belonging the fact level fl_n of $\mathcal{PG}$ and tainted with a set of*

taints $T_{fn} = \{t_1^{fn}, \cdot, t_m^{fn}\}$. *For any* $t_i^{fn} \in T$, *if* t_i^{fn} *is a ground taint, then one of the following conditions hold:*

1. fn *is an unsafe node regarding* C_i*;*
2. *There exists an unsafe node regarding* C_i fn_d *such that* fn_d *dominates* fn.

Proof. The proof is by contradiction: Let as assume that: (i) $\forall t_i^{fn} \in T_{fn} : t_i^{fn}$ is a ground taint, (ii) fn is a safe node regarding C_i, and (iii) $\neg(\exists fn_d \in \mathcal{PG}_t : unsafe\_node(fn_d, C_i) \wedge dominates(fn_d, fn))$. From (ii), we can deduce that the taint t_i^{fn} is a propagated taint as the node fn is a safe node regarding C_i.
Step A: based on (i) and the propagation rule (2), we can deduce that:

$$\exists! an_{n-1} \in al_{n-1} : fn \in \Sigma_{an_{n-1}} \tag{C.7}$$

(C.7) states that there exists only one action node an_{n-1} in the action level al_{n-1} that is providing fn. Since if there are many action nodes in the action level al_{n-1} that are providing fn, then the taint t_i^{fn} cannot be in any case a ground taint. Then from (C.7) and the propagation rule (2) we can deduce that:

$$t_i^{an_{n-1}} = t_i^{fn} \tag{C.8}$$

Step B: Based on the propagation rule (1) and (C.7), we can deduce that:

$$\exists! fn_{n-1} \in fl_{n-1} : fn_{n-1} \in \Delta_{an_{n-1}} \wedge \ t_i^{fn_{n-1}} = t_i^{fn} \tag{C.9}$$

From (C.7), (C.9) and Definition 38, we can deduce that:

$$dominates(fn_{n-1}, fn) \tag{C.10}$$

From (C.9) and (i) we deduce that: $t_i^{fn_{n-1}}$ is a ground taint (iv). Now, let us denote by $\Omega_{fn_{n-1}}$ the set of action nodes that are providing fn_{n-1}. Then, based on taint propagation rule (2) we deduce that:

$$t_i^{fn_{n-1}} = t_i^{fn_{n-1}} \wedge (\bigvee_{an \in \Omega_{fn_{n-1}}}) \tag{C.11}$$

Then based on (iv) and C.11, either (a) fn_{n-1} is an unsafe node or (b) the $t_i^{fn_{n-1}}$ has been propagated. In the case of (a) and based on (C.10) we can deduce a contradiction with (iii). In the case of (b), we will use the fact that: $\forall fn \in fl_1, \nexists an : an \in \Omega_{fn}$ to deduce that:

$$\exists fn_k, k \in [0, n-1[: t_i^{fn_k} = t_i^{fn} \wedge unsafe\_node(fn_k, C_i) \tag{C.12}$$

At this stage, by applying the steps A and B l times ($l = n - 1 - k$), we can deduce the following formulas:

$$\forall j \in [1, l],\ \exists! an_{n-1-j} \in al_{n-1-j} : an_{n-1-j} \in \Omega_{fn_{n-j}} \tag{C.13}$$

$$\forall j \in [1, l],\ \exists! fn_{n-1-j} \in fl_{n-1-j} : fn_{n-1-j} \in \Delta_{an_{n-1-j}} \wedge\ t_i^{fn_{n-1-j}} = t_i^{fn} \tag{C.14}$$

$$\forall j \in [1, l] :\ dominates(fn_{n-1-j}, fn_{n-j}) \tag{C.15}$$

From (C.12) and (C.14), we deduce that: $fn_{n-1-j} = fn_k \wedge unsafe\_node(fn_{n-1-j}, C_i)$ (v). Then, from (C.10) and (C.15) we can deduce that: $dominates(fn_{n-1-j}, fn)$ (vi). Finally, from (iii), (v) and (vi) we can deduce a contradiction. □

Lemma 4. *Given a planning graph $\mathcal{PG}$ and a fact node fn belonging to the fact level $fl_m \in \mathcal{PG}$. The following condition holds:*

$$\forall t : \Big(ground\_taint(t)\ \wedge\ (t_i^{fn} \models t)\Big) \rightarrow \Big((t_i^{fn} = t)\ \vee \\ (\exists fn', \exists j : j < m \wedge fn' \in fl_j \wedge t_i^{fn'} = t \wedge dominates(fn', fn))\Big)$$

Proof. Proof is by contradiction. Suppose that $\forall t$:

$$ground\_taint(t) \wedge t_i^{fn} \models t \tag{C.16}$$

$$t_i^{fn} \neq t \tag{C.17}$$

$$\neg(\exists fn', \exists j : j < m \wedge fn' \in fl_j \wedge t_i^{fn'} = t \wedge dominates(fn', fn)) \tag{C.18}$$

Case 1: (m=1). In this fist case, by assumption we have $fn \in fl_1$. Or by definition, the fact nodes in the fist fact level fl_1 represent the initial state of the target system and they are not produced by any action nodes in the planning graph. Then we deduce that:

$$\nexists an : an \in \mathcal{PG} \wedge fn \in \Sigma_{an} \tag{C.19}$$

Then based on the propagation rule (2), we deduce that $t_i^{fn} \models t$, and based on the fact that t is a ground taint, we deduce that $t_i^{fn} = t$ which is contradictory with (C.17).

Case 1: ($m \geq 1$).

Step A (begin): In this part of the proof, for the sake of clarity, let us denote Ω_{fn} the set of action nodes that provide fn ($\Omega_{fn} = \{an | fn \in \Sigma an\}$). By applying the taint propagation rule 2, we have:

$$t_i^{fn} = (t_i^{fn} \wedge (\bigvee_{an \in \Omega_{fn}} t_i^{an}) \tag{C.20}$$

Then based on (C.16) and (C.20) we deduce that $t_i^{fn} = t$ or $(\bigvee_{an \in \Omega_{fn}} t_i^{an}) \models t$.

Case 1.1: $t_i^{fn} = t$. we deduce a contradiction with (C.17).

Case 1.2: $(\bigvee_{an \in \Omega_{fn}} t_i^{an}) \models t$. By definition, Ω_{fn} represents the set of action nodes that provide fn. Then if we suppose that $\mathcal{PL} = \{\mathcal{P}_1 = \{p_1^1, \cdots, p_{m-1}^1\}, \cdots, \mathcal{P}_n = \{p_1^n, \cdots, p_{m-1}^n\}\}$ is the set of all parallel plans that correctly provide fn, then we have the following:

$$\forall \mathcal{P}_l \in \mathcal{PG} : (\bigwedge_{an \in p_{m-1}^l} t_i^{an}) \models t. \tag{C.21}$$

Using the taint propagation rule (1) together with (C.21), we continue to get:

$$\forall \mathcal{P}_l \in \mathcal{PG} : \Big(\bigwedge_{an \in p_{m-1}^l} (\bigwedge_{fn^{m-1} \in \Delta_{an}} t_i^{fn^{m-1}})\Big) \models t. \tag{C.22}$$

Then, based on the Definition 38 and (C.22), we can deduce that:

$$\exists fn^{m-1} : fn^{m-1} \in fl_{m-1} \ \wedge \ (t_i^{fn^{m-1}} \models t) \ \wedge \ dominates(fn^{m-1}, fn) \tag{C.23}$$

Step A (end).

Now, based on the fact that $\forall fn \in fl_1 : \Omega_{fn} = \emptyset$, by repeating the "Step A" $m-1$ times, we deduce that:

$$\begin{aligned} \exists k \in [1, m-1], \exists fn^{m-k}, \exists fn^{m-k+1}, \cdots, \exists fn^{m-1} : \Big(\big(\bigwedge_{j=m-k}^{m} (fn^j \in fl_j)\big) \\ \wedge \big((\bigwedge_{j=m-k}^{m} t_i^{fn^j}) \models t\big) \wedge \big(\bigwedge_{j=m-k}^{m} dominates(fn^{j-1}, fn^j)\big) \ \wedge \ t_i^{fn^{m-k}} = t\Big) \end{aligned} \tag{C.24}$$

Then we get:

$$\begin{aligned} (\bigwedge_{j=m-k}^{m-1} dominates(fn^{j-1}, fn^j)) \wedge dominates(fn^{m-1}, fn) \\ \xrightarrow{Lemma2} \quad dominates(fn^{m-k}, fn) \end{aligned} \tag{C.25}$$

Finally, from (C.24) and (C.25), we get:

$$\exists k \in [1, m-1], \exists fn^k \in fl_k : dominates(fn^k, fn) \wedge t_i^{fn^k} = t \tag{C.26}$$

which is contradictory with (C.18). □

Lemma 5. *Given a planning graph $\mathcal{PG}$ composed of n fact levels $fl_1, \cdots, fl_n$ and two fact nodes fn_1 and fn_2 belonging to $\mathcal{PG}$ and tainted respectively using $T_{fn_1} = \{t_1^{fn_1}, \cdots, t_l^{fn_1}\}$ and $T_{fn_2} = \{t_1^{fn_2}, \cdots, t_l^{fn_2}\}$. The following condition holds:*

$$\begin{aligned} \forall k \in [1, l], \forall l, m \in [1, n] : l < m \ \wedge \ fn_1 \in fl_l \ \wedge \ fn_2 \in fl_m \ \wedge \\ dominates(fn_1, fn_2) \rightarrow t_k^{fn_2} \models t_k^{fn_1}. \end{aligned}$$

Proof. Proof is by induction. Using Definition 38, fn_1 dominates fn_2 which allows us to get:

$$\forall w \in [1,q], \exists p, \exists an : p \in \mathcal{P}_w \ \wedge \ an \in p \ \wedge \ fn_1 \in \Delta_{an} \tag{C.27}$$

Where $\{\mathcal{P}_1 = \{p_1^1, \cdots, p_l^1\}, \cdots, \mathcal{P}_q = \{p_1^q, \cdots, p_l^q\}\}$ is the set of parallel plans that correctly provide fn_2. Then, using the propagation rule 1 we deduce that:

$$\forall k \in [1,l], \forall w \in [1,q], \exists p, \exists an : p \in \mathcal{P}_w \ \wedge \ an \in p \ \wedge \ t_k^{an} \models t_k^{fn_1} \tag{C.28}$$

Now let us demonstrate by recurrence that:

$$\begin{aligned} \forall k \in [1,l], \forall w \in [1,q], \forall i \in [1,m-1], \forall t : \Big(\big(\exists an : an \in p_i^w \ \wedge \ t_k^a n \models t \big) \\ \rightarrow \big(\exists an' : an' \in p_{m-1}^w \wedge t_k^{an'} \models t \big) \Big) \end{aligned} \tag{C.29}$$

Based on the propagation rule 2, we get:

$$\forall k \in [1,l], \forall fn, \forall an : \big(fn \in \Sigma_{an} \ \wedge \ (t_k^{an} \models t) \big) \rightarrow (t_k^{fn} \models t)$$

We continue using Definition 30 ((*iii*) and (*iv*)) to get:

$$\begin{aligned} \forall k \in [1,l], \forall w \in [1,q], \forall t : \Big(\big(\exists an : an \in p_i^w \ \wedge \ t_k^a n \models t \big) \\ \rightarrow \big(\exists an' : an' \in p_{i+1}^w \wedge t_k^{an'} \models t \big) \Big) \end{aligned}$$

Let us suppose that : $\forall k \in [1,k], \exists an^{m-2} : an^{m-2} \in p_{m-2}^w \wedge t_k^{an^{m-2}} \models t$, and based on the propagation rule 2 we deduce that:

$$\forall k \in [1,l], \forall fn : (fn \in \Sigma_{an^{m-2}}) \ \wedge \ (t_k^{an^{m-2}} \models t) \rightarrow (t_k^{fn} \models t)$$

Then by using Definition 30 ((*iii*) and (*iv*)) to get:

$$\forall k \in [1,l], \exists an' : (an \in p_{m-1}^w) \ \wedge \ (t_k^{an'} \models t)$$

which prove the correctness of (C.29). Finally, by using Definition 30 ((*v*) and (*vi*)), we deduce that:

$$\forall k \in [1,l] : t_k^{fn_2} \models t_k^{fn_1}$$

□

Now we can prove Theorem 12 as following.

Proof.

(1) Proof is by contradiction. Let us assume that:

1. $\big(\bigwedge_{j=1}^{n} t_i^{fn_j}) \models t\big) \ \wedge \ vio\_taint(t, C_i)$
2. $\exists \mathcal{P}_i^N \in \mathcal{PL}^N$ that correctly provides $fn_1, \cdots, fn_n$ without violating C_i

Based on Definition 44, we get:

$$\begin{aligned}\Big(\exists t : t \wedge vio\_taint(t, C_i)\Big) \rightarrow \Big(\exists fl, \exists! t_i^{fn'_1}, \cdots, \exists! t_i^{fn'_q} : fl \in \mathcal{PG} \ \wedge \\ \Big(\bigwedge_{j=1}^{q}(fn'_j \in fl)\Big) \ \wedge \ \Big((\bigwedge_{j=1}^{q} fn'_j) \rightarrow \neg C_i\Big) \ \wedge \ \Big((\bigwedge_{j=1}^{q} t_i^{fn'_j}) \rightarrow \neg t\Big) \\ \wedge \ \Big(\bigwedge_{j=1}^{q} ground\_taint(t_i^{fn'_j})\Big)\Big)\end{aligned} \tag{C.30}$$

Then from (1) and (C.30) we can deduce:

$$(\bigwedge_{j=1}^{n} t_i^{fn_j}) \models \bigwedge_{k=1}^{q}(t_i^{fn'_q} \ \wedge \ ground\_taint(t_i^{fn'_q})) \tag{C.31}$$

Then from (C.31) we get:

$$\forall k \in [1,q], \exists j \in [1,n] : \Big(\Big(t_i^{fn_j} \ \models \ t_i^{fn'_k}\Big) \ \wedge \ ground\_taint(t_i^{fn'_k})\Big) \tag{C.32}$$

Now, by applying Lemma 4 over (C.32) we get:

$$\begin{aligned}\forall k \in [1,q], \exists j \in [1,n] : \Big(\Big(t_i^{fn_j} = t_i^{fn'_k}\Big) \ \vee \ \Big(\exists fn'', \exists p : \ p < m \ \wedge \\ fn'' \in fl_p \ \wedge \ t_i^{fn''} = t_i^{fn'_k} \ \wedge \ dominates(fn'', fn_j)\Big)\Big)\end{aligned} \tag{C.33}$$

From (C.33), we can see that we have two cases:
Case 1: $\exists k \in [1,q], \exists j \in [1,n] : t_i^{fn_j} = t_i^{fn'_k}$. By definition, $t_i^{fn'_k}$ is a unique ground taint. Then, we deduce that $fn_j = fn'_k$. Then based on (C.30) we continue to get:

$$\Big(\bigwedge_{k=1}^{q}(fn'_k \in fl_m)\Big) \ \wedge \ \Big(\bigwedge_{k=1}^{q}(t_i^{fn'_k}) \rightarrow t\Big) \tag{C.34}$$

We can then deduce that $\{fn'_1, \cdots, fn'q\} \subseteq \{fn_1, \cdots, fn_n\}$. Finally, based on the fact that $(\bigwedge_{k=1}^{q} fn'_k) \rightarrow \neg C_i)$, then we can deduce that there is no parallel plan that could provide the set of nodes $fn_1, \cdots, fn_n$ without violating the constraint C_i, which is contradictory with (2).
Case 2:

$$\begin{aligned}\forall k \in [1,q], \exists j \in [1,n] : \ \Big(\exists fn'', \exists p : \ p < m \ \wedge \ fn'' \in fl_p \ \wedge \ t_i^{fn''} = t_i^{fn'_k} \\ \wedge \ dominates(fn'', fn_j)\Big)\end{aligned} \tag{C.35}$$

Let us suppose that $\mathcal{PL}^j = \{P_1^j = \{p_1^{1,j}, \cdots, p_{m-1}^{1,j}\}, \cdots, P_{r_j}^j = \{p_1^{r_j,j}, \cdots, p_{m-1}^{r_j,j}\}\}$ is the set of r_j correct plans that provide each $fn_j \in N$. Then by considering that $t_i^{fn''}$ and $t_i^{fn'_k}$ are unique and based on (C.35), we get the following:

$$\begin{aligned}\forall k \in [1,q], \exists j \in [1,n], \exists s_k \in [1,m-1], \forall v \in [1,r_j], \exists an : \mathcal{P}_v^j \in \mathcal{PL}^j \wedge \\ p_{s_k}^{v,j} \in \mathcal{P}_v^j \wedge an \in p_{s_k}^{v,j} \wedge fn'_k \in \Delta_{an}\end{aligned} \tag{C.36}$$

Based on (C.30) we have: $\bigwedge_{k=1}^{q}(fn'_j \in fl)$. Then we are able to deduce that $s_1 = s_2 = \cdots = s_q$ in (C.36). Now, let us suppose that $\mathcal{PL}^N = \{\mathcal{P}_1^N = \{p_1^{1,N}, \cdots, p_{m-1}^{1,N}\}, \cdots, \mathcal{P}_r^N = \{p_1^{r,N}, \cdots, p_{m-1}^{r,N}\}\}$ represents the set of all parallel plans that correctly provide N. Then by using the Theorem 1, we get the following:

$$\forall w \in [1,r], \forall s_k \in [1,m-1], \forall j \in [1,n], \exists v \in [1,r_j] : p_{s_k}^{w,N} = (\bigcup_{j=1}^{n} p_{s_k}^{v,j}) \tag{C.37}$$

From (C.36) and (C.37) we get:

$$\forall w \in [1,r], \exists s_k \in [1,m-1] : \Big(\bigwedge_{k=1}^{q} \Big(\exists an_j : an_j \in p_{s_k}^{w,N} \wedge fn'_k \in \Delta_{an}\Big)\Big) \tag{C.38}$$

Finally, from (C.30) and (C.38), we can deduce that there is no parallel plan that could provide the set of nodes $fn_1, \cdots, fn_n$ without violating the constraint C_i, which is contradictory with (2).

(2) Proof is by contradiction. Let us assume that:

1. $\forall i \in [1,l], \nexists t : \Big((\bigwedge_{j=1}^{n} t_i^{fn_j}) \models t\Big) \wedge vio\_taint(t, C_i)$
2. $\nexists \mathcal{P}_s^N \in \mathcal{PL}^N$ that correctly provides $fn_1, \cdots, fn_n$ without violating C_i

First let us suppose that $\mathcal{P}_w^N = \{p_1^{w,N}, \cdots, p_{m-1}^{w,N}\}, 1 \leq w \leq r$. Based on the fact that each ground literal used to specify the target system is represented by a fact node in $\mathcal{PG}$. Then, based on Definition 39 and the assumption (2) we get:

$$\begin{aligned}\forall w \in [1,r], \exists s_w \in [1,m-1], \exists fn_1^w, \exists fn_2^w, \cdots, \exists fn_q^w : \Big(\\ \Big((\bigcup_{j=1}^{q} fn_j^w) \subseteq (\bigcup_{an \in p_{sw}^{w,N}} \Delta_{an})\Big) \wedge \Big((\bigwedge_{j=1}^{q} fn_j^w) \rightarrow \neg C_i\Big)\Big)\end{aligned} \tag{C.39}$$

Based on the fact that each ground literal used to represent the target system is represented by a fact node in $\mathcal{PG}$, we can use Definition 39 to deduce that: $\{fn_1^1, \cdots, fn_q^1\} = \{fn_1^2, \cdots, fn_q^2\} = \cdots = \{fn_1^r, \cdots, fn_q^r\}$. Now let us suppose that:

$\forall i \in [1,n] : \mathcal{PL}^i = \{\mathcal{P}_1^i = \{p_1^{1,i}, \cdots, p_{m-1}^{1,i}\}, \cdots, \mathcal{P}_{r_i}^i = \{p_1^{r_i,i}, \cdots, p_{m-1}^{r_i,i}\}\}$, where $\mathcal{PL}^i$ is the set of all parallel plans that correctly provides $fn_i \in N$. We continue using Theorem 1 to get the following:

$$\forall j \in [1,n], \forall v_j \in [1,r_j], \forall s \in [1,m-1], \exists k \in [1,q] : (\bigcup_{j=1}^{n} p_s^{v_j,j}) = p_s^{k,N} \tag{C.40}$$

Then from (C.39) and (C.40), we get:

$$\begin{aligned} \exists s \in [1,m-1], \forall v_1 \in [1,r_1], \cdots, \forall v_n \in [1,r_n], \exists fn'_1, \cdots, \exists fn'_q : \\ \Big(\big(\bigcup_{k=1}^{q} fn'_k \big) \subseteq \big(\bigcup_{j=1}^{n} \bigcup_{an \in p_s^{v_j,j}} \Delta_{an} \big) \wedge \big((\bigwedge_{k=1}^{q} fn'_k) \rightarrow \neg C_i \big) \Big) \end{aligned} \tag{C.41}$$

Then, based on C.41 we continue to get:

$$\forall k \in [1,q], \exists s \in [1,m-1], \exists j \in [1,n], \forall v_j \in [1,r_j] : fn'_k \in \bigcup_{an \in p_s^{v_j,j}} \Delta_{an} \tag{C.42}$$

Then, using Definition 38 we can deduce that:

$$\forall k \in [1,q], \exists s \in [1,m-1], \exists j \in [1,n] : dominates(fn'_k, fn_j) \tag{C.43}$$

By supposing that each fn'_k is tainted using the taint $T_{fn_k} = \{t_1^{fn_k}, \cdots, t_l^{fn_k}\}$. Then, based on Lemma 5 we get:

$$\forall k \in [1,q], \forall i \in [1,l], \exists s \in [1,m-1], \exists j \in [1,n] : t_i^{fn_j} \models t_i^{fn'_k} \tag{C.44}$$

Then, based on Definition 44 we get:

$$\begin{aligned} \forall fn_1, \cdots, \forall fn_q : \big((\bigwedge_{k=1}^{q} fn'_k) \rightarrow \neg C_i \big) \rightarrow \big(\exists t : ((\bigwedge_{k=1}^{q} t_i^{fn_q}) \rightarrow t) \\ \wedge \ vio\_taint(t, C_i) \big) \end{aligned} \tag{C.45}$$

Then, using (C.41), (C.44) and (C.45) we deduce that:

$$\exists j_1, \cdots, j_q \in [1,n], \exists t : \big((\bigwedge_{k=1}^{q} t_i^{fn_k}) \models t \big) \ \wedge \ vio\_taint(t, C_i). \tag{C.46}$$

Finally, from C.46 we deduce that:

$$\exists t : \big((\bigwedge_{k=1}^{n} t_i^{fn_k}) \models t \big) \ \wedge \ vio\_taint(t, C_i). \tag{C.47}$$

Which is contradictory with the assumption (1). □

C.5 Proof of Theorem 13

We prove the previous theorem by a reduction from the NP-hard problem of cutting-stock problem [Garey and Johnson 1990], which is formulated as follows: *Cutting standard-sized pieces of stock material into a set pieces of specified sizes while minimizing material wasted.* We define the correspondence between the problem of finding the best compromise between satisfying a set of goals and ensuring a set security constraints and cutting-stock problem as follows. Let us suppose that an autonomous system aims to satisfy a set of goals $\mathcal{G} = \{G_1, \cdots, G_n\}$ while ensuring a set of security constraints $\mathcal{C} = \{C_1, \cdots, C_m\}$. Any piece corresponds to a goal $G_i \in \mathcal{G}$ and any piece of wasted material corresponds to a security constraint $C_j \in \mathcal{C}$. So, any algorithm finding the best compromise between satisfying specified goals and ensuring security constraints (Definition 45) can be used to solve the cutting-stock problem to optimality.

List of Publications

International Journals

- A. Bkakria, F. Cuppens, N. Cuppens-Boulahia, J. M. Fernandez, and D. Gross-Amblard. Preserving multi-relational outsourced databases confidentiality using fragmentation and encryption. JoWUA, 4(2):39–62, 2013.

International Conferences

- A. Bkakria, F. Cuppens, N. Cuppens-Boulahia, and D. Gross-Amblard. Security mechanisms planning to enforce security policies. Foundations and Practice of Security - 7th International Symposium, FPS 2015, 26-28 October, Clermont-Ferrand, France, 2015.
- A. Bkakria, F. Cuppens, N. Cuppens-Boulahia, and D. Gross-Amblard. Specification and deployment of integrated security policies for outsourced data. In Data and Applications Security and Privacy XXVIII - 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings, pages 17–32, 2014.
- A. Bkakria, A. Schaad, F. Kerschbaum, F. Cuppens, N. Cuppens-Boulahia, and D. Gross-Amblard. Optimized and controlled provisioning of encrypted outsourced data. In 19th ACM Symposium on Access Control Models and Technologies, SACMAT '14, London, ON, Canada - June 25 - 27, 2014, pages 141–152, 2014.
- A. Bkakria, F. Cuppens, N. Cuppens-Boulahia, and J. M. Fernandez. Confidentiality-preserving query execution of fragmented outsourced data. In Information and Communicatiaon Technology - International Conference, ICT-EurAsia 2013, Yogyakarta, Indonesia, March 25-29, 2013. Proceedings, pages 426–440, 2013.

Bibliography

[Doe] Transaction processing performance council. `http://www.tpc.org/`. Accessed: 2015-07-22. 67

[DBL 2007] *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*. IEEE Computer Society, 2007. 189, 196

[UE2 2014] European union agency for fundamental rights. handbook on european data protection law. `http://www.echr.coe.int/Documents/Handbook_data_protection_ENG.pdf`, 2014. Accessed: 2015-06-17. 4

[Aggarwal et al. 2005] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu. Two can keep A secret: A distributed architecture for secure database services. In *CIDR*, pages 186–199, 2005. 23

[Agrawal et al. 2004] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 563–574, 2004. 13, 16, 17, 56

[Aho et al. 1983] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983. 44

[Amanatidis et al. 2007] G. Amanatidis, A. Boldyreva, and A. O'Neill. Provably-secure schemes for basic query support in outsourced databases. In S. Barker and G. Ahn, editors, *Data and Applications Security XXI, 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Redondo Beach, CA, USA, July 8-11, 2007, Proceedings*, volume 4602 of *Lecture Notes in Computer Science*, pages 14–30, 2007. Springer. 13, 19

[Amazon] Amazon. Amazon's web services. `http://aws.amazon.com/`. Accessed: 2015-07-16. 2

[Amble and Knuth 1974] O. Amble and D. E. Knuth. Ordered hash tables. *Comput. J.*, 17(2):135–142, 1974. 44

[Armando et al. 2014] A. Armando, R. Carbone, and L. Compagna. SATMC: A sat-based model checker for security-critical systems. In E. Ábrahám and K. Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, volume 8413 of *Lecture Notes in Computer Science*, pages 31–45, 2014. Springer. 107

[Armando and Compagna 2002] A. Armando and L. Compagna. Automatic sat-compilation of protocol insecurity problems via reduction to planning. In D. Peled and M. Y. Vardi, editors, *Formal Techniques for Networked and Distributed Systems - FORTE 2002, 22nd IFIP WG 6.1 International Conference Houston, Texas, USA, November 11-14, 2002, Proceedings*, volume 2529 of *Lecture Notes in Computer Science*, pages 210–225, 2002. Springer. 107

[Armando et al. 2003] A. Armando, L. Compagna, and P. Ganty. Sat-based model-checking of security protocols using planning graph analysis. In K. Araki, S. Gnesi, and D. Mandrioli, editors, *FME 2003: Formal Methods, International Symposium of Formal Methods Europe, Pisa, Italy, September 8-14, 2003, Proceedings*, volume 2805 of *Lecture Notes in Computer Science*, pages 875–893, 2003. Springer. 107

[Bartoletti et al. 2009] M. Bartoletti, P. Degano, and G. L. Ferrari. Planning and verifying service composition. *Journal of Computer Security*, 17(5):799–837, 2009. 107

[Bellare et al. 2007] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In A. Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552, 2007. Springer. 58

[Bellare et al. 1997] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 394–403, 1997. IEEE Computer Society. 37

[Bellare et al. 2008] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without ran-

dom oracles. In D. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 360–378, 2008. Springer. 34

[Bertossi 1981] A. A. Bertossi. The edge hamiltonian path problem is np-complete. *Inf. Process. Lett.*, 13(4/5):157–159, 1981. 127

[Bhattacharya and Cortesi 2009] S. Bhattacharya and A. Cortesi. A distortion free watermark framework for relational databases. In *ICSOFT 2009 - Proceedings of the 4th International Conference on Software and Data Technologies, Volume 2, Sofia, Bulgaria, July 26-29, 2009*, pages 229–234, 2009. 97

[Bingmann 2008] T. Bingmann. Stx b+ tree c++ template classes v 0.8.3. `http://panthema.net/2007/stx-btree/`, 2008. 50

[Biskup et al. 2011] J. Biskup, M. Preuss, and L. Wiese. On the inference-proofness of database fragmentation satisfying confidentiality constraints. In X. Lai, J. Zhou, and H. Li, editors, *Information Security, 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings*, volume 7001 of *Lecture Notes in Computer Science*, pages 246–261, 2011. Springer. 30

[Bkakria et al. 2013a] A. Bkakria, F. Cuppens, N. Cuppens-Boulahia, and J. M. Fernandez. Confidentiality-preserving query execution of fragmented outsourced data. In *Information and Communicatiaon Technology - International Conference, ICT-EurAsia 2013, Yogyakarta, Indonesia, March 25-29, 2013. Proceedings*, pages 426–440, 2013. 5, 147, 152

[Bkakria et al. 2013b] A. Bkakria, F. Cuppens, N. Cuppens-Boulahia, J. M. Fernandez, and D. Gross-Amblard. Preserving multi-relational outsourced databases confidentiality using fragmentation and encryption. *JoWUA*, 4(2):39–62, 2013. 5, 81, 87, 97, 102, 147, 152

[Bkakria et al. 2014a] A. Bkakria, F. Cuppens, N. Cuppens-Boulahia, and D. Gross-Amblard. Specification and deployment of integrated security policies for outsourced data. In *Data and Applications Security and Privacy XXVIII - 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings*, pages 17–32, 2014. 6, 82, 152

[Bkakria et al. 2014b] A. Bkakria, A. Schaad, F. Kerschbaum, F. Cuppens, N. Cuppens-Boulahia, and D. Gross-Amblard. Optimized and controlled provisioning of encrypted outsourced data. In *19th ACM Symposium on Access*

Control Models and Technologies, SACMAT '14, London, ON, Canada - June 25 - 27, 2014, pages 141–152, 2014. 5, 147, 152

[Bloom 1970] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970. 18

[Blum and Furst 1995] A. Blum and M. L. Furst. Fast planning through planning graph analysis. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 1636–1642, 1995. 116

[Blum and Furst 1997] A. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artif. Intell.*, 90(1-2):281–300, 1997. 116

[Boho et al. 2013] A. Boho, G. V. Wallendael, A. Dooms, J. D. Cock, G. Braeckman, P. Schelkens, B. Preneel, and de R. V. Walle. End-to-end security for video distribution: The combination of encryption, watermarking, and video adaptation. *IEEE Signal Process. Mag.*, 30(2):97–107, 2013. 106

[Boldyreva et al. 2009] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 224–241, 2009. 13, 17, 58, 69, 96, 97, 113, 126

[Boldyreva et al. 2011] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In [Rogaway 2011], pages 578–595. 13, 17, 58, 69

[Boldyreva et al. 2012] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. *IACR Cryptology ePrint Archive*, 2012:625, 2012. 13, 17

[Boneh 1998] D. Boneh. The decision diffie-hellman problem. In J. Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63, 1998. Springer. 18

[Boneh et al. 2005] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In *Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341, 2005. Springer. 13, 21

[Boneh and Waters 2007] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 535–554, 2007. 16

[Botelho et al. 2007] F. C. Botelho, R. Pagh, and N. Ziviani. Simple and space-efficient minimal perfect hash functions. In F. K. H. A. Dehne, J. Sack, and N. Zeh, editors, *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, volume 4619 of *Lecture Notes in Computer Science*, pages 139–150, 2007. Springer. 48

[Botelho and Zivian 2007] F. C. Botelho and N. Zivian. Cmph : C minimal perfect hashing library. `http://cmph.sourceforge.net/`, 2007. 50

[Botelho and Ziviani 2007] F. C. Botelho and N. Ziviani. External perfect hashing for very large key sets. In M. J. Silva, A. H. F. Laender, R. A. Baeza-Yates, D. L. McGuinness, B. Olstad, Ø. H. Olsen, and A. O. Falcão, editors, *CIKM*, pages 653–662, 2007. ACM. 50

[Brakerski et al. 2012] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In S. Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 309–325, 2012. ACM. 21

[Brakerski and Vaikuntanathan 2011a] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:109, 2011. 21

[Brakerski and Vaikuntanathan 2011b] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In [Rogaway 2011], pages 505–524. 21

[Cancellaro et al. 2011] M. Cancellaro, F. Battisti, M. Carli, G. Boato, F. G. B. D. Natale, and A. Neri. A commutative digital image watermarking and encryption method in the tree structured haar transform domain. *Sig. Proc.: Image Comm.*, 26(1):1–12, 2011. 106

[Chan et al. 2012] P. P. F. Chan, L. C. K. Hui, and S. Yiu. Droidchecker: analyzing android applications for capability leak. In M. Krunz, L. Lazos, R. D. Pietro, and W. Trappe, editors, *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WISEC 2012, Tucson, AZ, USA, April 16-18, 2012*, pages 125–136, 2012. ACM. 135

[Charpentier et al. 2011] A. Charpentier, C. Fontaine, T. Furon, and I. J. Cox. An asymmetric fingerprinting scheme based on tardos codes. In *Information Hiding - 13th International Conference, IH 2011, Prague, Czech Republic, May 18-20, 2011, Revised Selected Papers*, pages 43–58, 2011. 97

[Chase and Kamara 2011] M. Chase and S. Kamara. Structured encryption and controlled disclosure. *IACR Cryptology ePrint Archive*, 2011:10, 2011. 13, 19

[Choi et al. 2015] Y. Choi, M. Park, J. Eom, and T. Chung. Dynamic binary analyzer for scanning vulnerabilities with taint analysis. *Multimedia Tools Appl.*, 74(7):2301–2320, 2015. 135

[Chor et al. 1998] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. *IACR Cryptology ePrint Archive*, 1998:3, 1998. 29, 43

[Ciriani et al. 2007] V. Ciriani, di S. D. C. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Fragmentation and encryption to enforce privacy in data storage. In *Computer Security - ESORICS 2007, 12th European Symposium On Research In Computer Security, Dresden, Germany, September 24-26, 2007, Proceedings*, pages 171–186, 2007. 5, 23, 24, 25, 28, 31, 32, 35, 97, 106, 147, 152, 153

[Ciriani et al. 2009] V. Ciriani, di S. D. C. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Fragmentation design for efficient query execution over sensitive distributed databases. In *29th IEEE International Conference on Distributed Computing Systems (ICDCS 2009), 22-26 June 2009, Montreal, Québec, Canada*, pages 32–39, 2009. 5, 23, 24, 25, 28, 31, 147, 152, 153

[Ciriani et al. 2012] V. Ciriani, di S. D. C. Vimercati, S. Foresti, G. Livraga, and P. Samarati. An OBDD approach to enforce confidentiality and visibility constraints in data publishing. *Journal of Computer Security*, 20(5):463–508, 2012. 24, 32

[CloudMe] CloudMe. Cloudme. `https://www.cloudme.com/en`. Accessed: 2015-06-16. 1

[Cohn 2000] P. M. Cohn. *Introduction to Ring Theory*. Springer Science & Business Media, 2000. 20

[Curtmola et al. 2006] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In A. Juels, R. N. Wright, and di S. D. C. Vimercati, editors, *Proceedings of*

the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006, pages 79–88, 2006. ACM. 13, 19

[Dai 1995] W. Dai. The crypto++ library. `http://www.cryptopp.com/`, 1995. 50

[Damiani et al. 2003a] E. Damiani, di S. D. C. Vimercati, M. Finetti, S. Paraboschi, P. Samarati, and S. Jajodia. Implementation of a storage mechanism for untrusted dbmss. In *2nd International IEEE Security in Storage Workshop (SISW 2003), Information Assurance, The Storage Security Perspective, 31 October 2003, Washington, DC, USA*, pages 38–46, 2003. 13, 15

[Damiani et al. 2003b] E. Damiani, di S. D. C. Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational dbmss. In *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washington, DC, USA, October 27-30, 2003*, pages 93–102, 2003. 13, 15

[datalossdb 2015] datalossdb. Open security foundation. `http://datalossdb.org`, 2015. Accessed: 2015-09-16. 3

[Davida et al. 1981] G. I. Davida, D. L. Wells, and J. B. Kam. A database encryption system with subkeys. *ACM Trans. Database Syst.*, 6(2):312–328, 1981. 11

[Davis and McKim 1976] C. C. Davis and V. R. McKim. Temporal modalities and the future. *Notre Dame Journal of Formal Logic*, 17(2):233–238, 1976. 82

[di Vimercati et al. 2010] di S. D. C. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Fragments and loose associations: Respecting privacy in data publishing. *PVLDB*, 3(1):1370–1381, 2010. 24

[Dixon and Fisher 2000] C. Dixon and M. Fisher. Resolution-based proof for multimodal temporal logics of knowledge. In *Seventh International Workshop on Temporal Representation and Reasoning, TIME 2000, Nova Scotia, Canada, July 7-9, 2000*, pages 69–78, 2000. IEEE Computer Society. 83, 84

[Dropbox] Dropbox. Dropbox. `https://www.dropbox.com/`. Accessed: 2015-06-16. 1

[ElGamal 1985] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. 21

[Emerson 1995] E. A. Emerson. Temporal and modal logic. In *HANDBOOK OF THEORETICAL COMPUTER SCIENCE*, pages 995–1072, 1995. Elsevier. 85

[Enck et al. 2010] W. Enck, P. Gilbert, B. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In R. H. Arpaci-Dusseau and B. Chen, editors, *9th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2010, October 4-6, 2010, Vancouver, BC, Canada, Proceedings*, pages 393–407, 2010. USENIX Association. 135

[Fagin et al. 1995] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995. 86

[Fikes and Nilsson 1971] R. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208, 1971. 117, 161

[FIPS. 2001] N. FIPS. Announcing the advanced encryption standard (aes). *Processing Standards Publication*, 197, 2001. 12

[Frazzoli et al. 2000] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS*, 25:116–129, 2000. 134

[Fredman et al. 1984] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with 0(1) worst case access time. *J. ACM*, 31(3):538–544, 1984. 19

[Gabbay et al. 1980] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '80, pages 163–173, New York, NY, USA, 1980. ACM. 83

[Garey and Johnson 1990] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990. 63, 178

[Gentry 2009] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford, CA, USA, 2009. 13, 21

[Gentry et al. 2012] C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867, 2012. Springer. 22

[Gentry et al. 2013] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, 2013. Springer. 21

[Ghallab et al.] M. Ghallab, C. Knoblock, M. Veloso, D. Weld, and D. Wilkins. Pddl - the planning domain definition language. version 1.2. `http://homepages.inf.ed.ac.uk/mfourman/tools/propplan/pddl.pdf`. Accessed: 2015-08-05. 118

[GMail] GMail. Google's gmail. `https://www.gmail.com/`. Accessed: 2015-07-16. 2

[Goh 2003a] E. Goh. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003. 13, 18

[Goh 2003b] E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/`. 19

[Goldberg 2007a] I. Goldberg. Improving the robustness of private information retrieval. In [DBL 2007], pages 131–148. 50

[Goldberg 2007b] I. Goldberg. Percy++ / pir in c++. `http://percy.sourceforge.net/`, 2007. 50

[Goldreich 2004] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2004. 21

[Goldwasser and Micali 1982] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM. 21

[Golle et al. 2004] P. Golle, J. Staddon, and B. R. Waters. Secure conjunctive keyword search over encrypted data. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004, Proceedings*, volume 3089 of *Lecture Notes in Computer Science*, pages 31–45, 2004. Springer. 13, 18

[Google] Google. Google app engine. `https://appengine.google.com/`. Accessed: 2015-07-16. 2

[GoogleDrive] GoogleDrive. Google drive. https://drive.google.com/. Accessed: 2015-06-16. 1

[Grofig et al. 2014] P. Grofig, M. Härterich, I. Hang, F. Kerschbaum, M. Kohler, A. Schaad, A. Schröpfer, and W. Tighzert. Experiences and observations on the industrial implementation of a system to search over outsourced encrypted data. In S. Katzenbeisser, V. Lotz, and E. R. Weippl, editors, *Sicherheit 2014: Sicherheit, Schutz und Zuverlässigkeit, Beiträge der 7. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), 19.-21. März 2014, Wien, Österreich*, volume 228 of *LNI*, pages 115–125, 2014. GI. 55

[Gross-Amblard 2011] D. Gross-Amblard. Query-preserving watermarking of relational databases and xml documents. *ACM Trans. Database Syst.*, 36(1):3, 2011. 81

[Guo et al. 2006] F. Guo, J. Wang, and D. Li. Fingerprinting relational databases. In *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, 2006*, pages 487–492, 2006. 97

[Guttman 1984] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984*, pages 47–57, 1984. 16

[Hacigümüs et al. 2002] H. Hacigümüs, B. R. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002*, pages 216–227, 2002. 13, 30, 56, 81

[Halpern and Vardi 1989] J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time. i. lower bounds. *J. Comput. Syst. Sci.*, 38(1):195–237, 1989. 83

[Hintikka 1986] J. Hintikka. Reasoning about knowledge in philosophy: The paradigm of epistemic logic. In *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge, Monterey, CA, March 1986*, pages 63–80, 1986. 82

[Hore et al. 2004] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 720–731, 2004. 13, 14

[Huang et al. 2004] Y. Huang, F. Yu, C. Hang, C. Tsai, D. T. Lee, and S. Kuo. Verifying web applications using bounded model checking. In *2004 International Conference on Dependable Systems and Networks (DSN 2004), 28 June - 1 July 2004, Florence, Italy, Proceedings*, pages 199–208, 2004. IEEE Computer Society. 135

[Hudic et al. 2013] A. Hudic, S. Islam, P. Kieseberg, S. Rennert, and E. R. Weippl. Data confidentiality using fragmentation in cloud computing. *Int. J. Pervasive Computing and Communications*, 9(1), 2013. 22, 23

[Irwin et al. 2008] K. Irwin, T. Yu, and W. H. Winsborough. Avoiding information leakage in security-policy-aware planning. In V. Atluri and M. Winslett, editors, *Proceedings of the 2008 ACM Workshop on Privacy in the Electronic Society, WPES 2008, Alexandria, VA, USA, October 27, 2008*, pages 85–94, 2008. ACM. 107

[Islam et al. 2012] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012. The Internet Society. 58

[J. Gantz 2012] D. R. J. Gantz. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView*, pages 1–16, 2012. 1

[Jovanovic et al. 2010] N. Jovanovic, C. Kruegel, and E. Kirda. Static analysis for detecting taint-style vulnerabilities in web applications. *Journal of Computer Security*, 18(5):861–907, 2010. 135

[JSqlParser] JSqlParser. 75

[Kamara and Papamanthou 2013] S. Kamara and C. Papamanthou. Parallel and dynamic searchable symmetric encryption. In A. Sadeghi, editor, *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, volume 7859 of *Lecture Notes in Computer Science*, pages 258–274, 2013. Springer. 13, 20

[Kamara et al. 2012] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In T. Yu, G. Danezis, and V. D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 965–976, 2012. ACM. 13, 19

[Kautz and Selman 1996] H. A. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic and stochastic search. In *Proceedings of the Thirteenth*

National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 2., pages 1194–1201, 1996. 128

[Kautz and Selman 1999] H. A. Kautz and B. Selman. Unifying sat-based and graph-based planning. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 318–325, 1999. 7, 133

[Kautz et al. 2006] H. A. Kautz, B. Selman, and J. Hoffmann. SatPlan: Planning as satisfiability. In *Abstracts of the 5th International Planning Competition*, 2006. 129

[Kerschbaum et al. 2013a] F. Kerschbaum, P. Grofig, I. Hang, M. Härterich, M. Kohler, A. Schaad, A. Schröpfer, and W. Tighzert. Adjustably encrypted in-memory column-store. In A. Sadeghi, V. D. Gligor, and M. Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 1325–1328, 2013. ACM. 58

[Kerschbaum et al. 2013b] F. Kerschbaum, M. Härterich, P. Grofig, M. Kohler, A. Schaad, A. Schröpfer, and W. Tighzert. Optimal re-encryption strategy for joins in encrypted databases. In L. Wang and B. Shafiq, editors, *Data and Applications Security and Privacy XXVII - 27th Annual IFIP WG 11.3 Conference, DBSec 2013, Newark, NJ, USA, July 15-17, 2013. Proceedings*, volume 7964 of *Lecture Notes in Computer Science*, pages 195–210, 2013. Springer. 58

[Kerschbaum et al. 2013c] F. Kerschbaum, M. Härterich, M. Kohler, I. Hang, A. Schaad, A. Schröpfer, and W. Tighzert. An encrypted in-memory column-store: The onion selection problem. In A. Bagchi and I. Ray, editors, *Information Systems Security - 9th International Conference, ICISS 2013, Kolkata, India, December 16-20, 2013. Proceedings*, volume 8303 of *Lecture Notes in Computer Science*, pages 14–26, 2013. Springer. 59

[Kindel et al. 2000] R. Kindel, D. Hsu, J. Latombe, and S. M. Rock. Kinodynamic motion planning amidst moving obstacles. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, April 24-28, 2000, San Francisco, CA, USA*, pages 537–543, 2000. IEEE. 134

[Kolesnikov and Shikfa 2012] V. Kolesnikov and A. Shikfa. On the limits of privacy provided by order-preserving encryption. *Bell Labs Technical Journal*, 17(3):135–146, 2012. 17

[Krishna et al. 2012] R. K. N. S. Krishna, T. J. V. R. K. M. K. Sayi, R. Mukkamala, and P. K. Baruah. Efficient privacy-preserving data distribution in outsourced environments: a fragmentation-based approach. In K. Gopalan and S. M. Thampi, editors, *2012 International Conference on Advances in Computing, Communications and Informatics, ICACCI '12, Chennai, India, August 3-5, 2012*, pages 589–595, 2012. ACM. 23

[Lewis and Langford 1932] C. I. Lewis and C. H. Langford. *Symbolic logic, by Clarence Irving Lewis and Cooper Harold Langford*. Century Co New York, 1932. 83

[Lewis 2014] D. Lewis. icloud data breach: Hacking and celebrity photos. `http://www.forbes.com/sites/davelewis/2014/09/02/icloud-data-breach-hacking-and-nude-celebrity-photos/`, 2014. Accessed: 2015-06-16. 3

[Li et al. 2007] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 106–115, 2007. 81, 97

[Machanavajjhala et al. 2006] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, page 24, 2006. 81, 97

[Maddock 2001] J. Maddock. Regular expressions in c++. `http://www.boost.org/`, 2001. 49

[Mell and Grance 2011] P. Mell and T. Grance. The nist definition of cloud computing. *National Institute of Standards and Technology*, 800(145), 2011. 2

[Microsoft] Microsoft. Microsoft share point. `https://products.office.com/fr-fr/sharepoint/collaboration`. Accessed: 2015-07-16. 2

[Moataz and Shikfa 2013] T. Moataz and A. Shikfa. Boolean symmetric searchable encryption. In *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*, pages 265–276, 2013. 96, 97

[Mulazzani et al. 2011] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. R. Weippl. Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*, 2011. 3

[Mykletun et al. 2004] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2004, San Diego, California, USA*, 2004. 81

[Narasimha and Tsudik 2005] M. Narasimha and G. Tsudik. DSAC: an approach to ensure integrity of outsourced databases using signature aggregation and chaining. *IACR Cryptology ePrint Archive*, 2005:297, 2005. 81

[Navathe et al. 1984] S. B. Navathe, S. Ceri, G. Wiederhold, and J. Dou. Vertical partitioning algorithms for database design. *ACM Trans. Database Syst.*, 9(4):680–710, 1984. 28

[Newton 2011] D. Newton. Dropbox authentication: insecure by design. `http://dereknewton.com/2011/04/dropbox-authentication-static-host-ids`, 2011. Accessed: 2015-06-16. 3

[NIST 1985] NIST. Trusted computer system evaluation criteria. `http://csrc.nist.gov/publications/history/dod85.pdf`, 1985. Accessed: 2015-06-17. 3

[NIST 1999] N. B. O. S. NIST. Data encryption standard (des). *Technology*, 46(3):1–26, 1999. 12

[NSA 2013a] NSA. Nsa spying poisons the cloud market: survey. `http://www.zdnet.com/nsa-spying-poisons-the-cloud-market-survey-7000022964/`, 2013. Accessed: 2015-07-06. 11

[NSA 2013b] NSA. Nsa surveillance: First prism, now muscled out of cloud. `http://www.darkreading.com/cloud-security/nsa-surveillance-first-prismnow-muscled-out-of-cloud/d/d-id/1112686`, 2013. Accessed: 2015-07-06. 11

[Oracle] Oracle. Mysql. `http://www.mysql.com/`. 50

[Paillier 1999] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 223–238, 1999. 13, 21, 58, 69, 97, 113, 126

[Park 2011] J. H. Park. Efficient hidden vector encryption for conjunctive queries on encrypted data. *IEEE Trans. Knowl. Data Eng.*, 23(10):1483–1497, 2011. 13, 16

[PCIDSS] PCIDSS. Pci security standards council. `https://www.pcisecuritystandards.org/`. Accessed: 2015-07-16. 59

[Pipatsrisawat and Darwiche 2007] Rsat 2.0: Sat solver description. Technical report, 2007. 128

[Pohlig and Hellman 1978] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978. 97

[Popa et al. 2011] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In T. Wobber and P. Druschel, editors, *Proceedings of the 23rd ACM Symposium on Operating Systems Principles 2011, SOSP 2011, Cascais, Portugal, October 23-26, 2011*, pages 85–100, 2011. ACM. 55, 106

[Pournaghshband 2008] V. Pournaghshband. A new watermarking approach for relational data. In *Proceedings of the 46th Annual Southeast Regional Conference, 2008, Auburn, Alabama, March 28-29, 2008*, pages 127–131, 2008. 97

[Pretschner et al. 2008] A. Pretschner, M. Hilty, D. A. Basin, C. Schaefer, and T. Walter. Mechanisms for usage control. In M. Abe and V. D. Gligor, editors, *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*, pages 240–244, 2008. ACM. 106

[RackSpace] RackSpace. Rackspace,Äôs cloud services. `http://www.rackspace.com/`. Accessed: 2015-07-16. 3

[Randell 1969] B. Randell. A note on storage fragmentation and program segmentation. *Commun. ACM*, 12(7):365–369, 1969. 22

[Rivest et al. 1978] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978. 20

[Robinson et al. 2008] N. Robinson, C. Gretton, and D.-N. Pham. Co-plan: Combining sat-based planning with forward-search. *Proc. IPC-6*, 2008. 128, 129

[Rogaway 2011] P. Rogaway, editor. *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*. Springer, 2011. 184, 185

[Rosser 1984] J. B. Rosser. Highlights of the history of the lambda-calculus. *IEEE Annals of the History of Computing*, 6(4):337–349, 1984. 107

[Salesforce] Salesforce. Salesforce.com. `http://www.salesforce.com`. Accessed: 2015-07-16. 2

[SAP] SAP. Seeed. `https://www.sics.se/sites/default/files/pub/andreasschaad.pdf`. Accessed: 2015-06-16. 78

[Shi et al. 2007] E. Shi, J. Bethencourt, H. T. Chan, D. X. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In [DBL 2007], pages 350–364. 13, 15, 16

[Sion 2008] R. Sion. Database watermarking for copyright protection. In *Handbook of Database Security - Applications and Trends*, pages 297–328. 2008. 81

[Song et al. 2000] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*, pages 44–55, 2000. IEEE Computer Society. 13, 18, 69

[Stehlé and Steinfeld 2010] D. Stehlé and R. Steinfeld. Faster fully homomorphic encryption. In M. Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 377–394, 2010. Springer. 21

[Sweeney 2002] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002. 81, 97, 98

[Szalas 1987] A. Szalas. A complete axiomatic characterization of first-order temporal logic of linear time. *Theor. Comput. Sci.*, 54:199–214, 1987. 87

[Tseng et al. 2013] F. Tseng, Y. Liu, R. Chen, and B. P. Lin. Statistics on encrypted cloud data. In *Advances in Information and Computer Security - 8th International Workshop on Security, IWSEC 2013, Okinawa, Japan, November 18-20, 2013, Proceedings*, pages 133–150, 2013. 97

[Vaikuntanathan 2011] V. Vaikuntanathan. Computing blindfolded: New developments in fully homomorphic encryption. In R. Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 5–16, 2011. IEEE Computer Society. 21

[van Dijk et al. 2010] van M. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In H. Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43, 2010. Springer. 21

[Verizon 2015] Verizon. 2015 data breach investigations report. `http://www.verizonenterprise.com/DBIR/`, 2015. Accessed: 2015-06-16. 3

[Wang et al. 2014] B. Wang, Y. Hou, M. Li, H. Wang, and H. Li. Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index. In *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 111–122, 2014. 13, 16

[Wang and Shieh 2015] C. Wang and S. W. Shieh. DROIT: dynamic alternation of dual-level tainting for malware analysis. *J. Inf. Sci. Eng.*, 31(1):111–129, 2015. 135

[Xiao and Yen 2012] L. Xiao and I. Yen. Security analysis for order preserving encryption schemes. In *46th Annual Conference on Information Sciences and Systems, CISS 2012, Princeton, NJ, USA, March 21-23, 2012*, pages 1–6, 2012. IEEE. 17, 58, 126

[Xie and Aiken 2006] Y. Xie and A. Aiken. Static detection of security vulnerabilities in scripting languages. In A. D. Keromytis, editor, *Proceedings of the 15th USENIX Security Symposium, Vancouver, BC, Canada, July 31 - August 4, 2006*, 2006. USENIX Association. 135

[YahooMail] YahooMail. Yahoo mail. `https://mail.yahoo.com/`. Accessed: 2015-07-16. 2

[Yoo et al. 2013] C. Yoo, R. Fitch, and S. Sukkarieh. Provably-correct stochastic motion planning with safety constraints. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pages 981–986, 2013. IEEE. 134

[Zhang et al. 2006] Y. Zhang, X. Niu, D. Zhao, J. Li, and S. Liu. Relational databases watermark technique based on content characteristic. In *First International Conference on Innovative Computing, Information and Control (ICICIC 2006), 30 August - 1 September 2006, Beijing, China*, pages 677–680, 2006. 97, 100

List of Figures

Résumé

Les avancées récentes dans les domaines de l'informatique en nuage sont en train de transformer la manière dont les informations sont gérées et consommées. Les fournisseurs de services de l'informatique en nuage sont de plus en plus tenus d'assumer leur responsabilité par rapport au stockage, ainsi que du partage efficace et fiable des données externalisées. Les propriétaires des données hésitent à accorder une confiance aveugle aux fournisseurs de services Cloud quant à la protection de leurs données externalisées. Ceci est dû au fait que, pour les récentes fuites de données externalisées, des vulnérabilités au sein des services Cloud ont été exploitées. Cela soulève alors la nécessité d'inventer de nouveaux modèles et de nouvelles approches permettant la définition et la mise en application des politiques de sécurité et de fonctionnalité sur les données externalisées. Cette dissertation vise à surmonter ce dilemme, tout en tenant compte de deux types de politiques de sécurité. En premier lieu, nous nous concentrons sur la spécification et le déploiement des politiques de confidentialité, qui nécessite : (1) la protection des données sensibles externalisées stockées dans des serveurs Cloud considérés comme étant non fiables, et (2) l'utilisation efficace des données externalisées par les utilisateurs autorisés. En deuxième lieu, nous abordons la problématique de la spécification et le déploiement des politiques de sécurité hétérogènes (par exemple, les politiques de confidentialité, de protection de la vie privée, d'intégrité, etc.) sur les données externalisées.

Mots-clés : Données Externalisées, Politique de sécurité, Politique de confidentialité, Spécification formelles, Méthode formelles

Abstract

Recent advance in cloud computing has transformed the way information is managed and consumed. Cloud service providers are increasingly required to take responsibility for the storage as well as the eDcient and reliable sharing of information, thus carrying out a «data outsourcing» architecture. Despite that outsourcing information on Cloud service providers may cut down data owners' responsibility of managing data while increasing data availability, data owners hesitate to fully trust Cloud service providers to protect their outsourced data. Recent data breaches on Cloud storage providers have exacerbated these security concerns. In response, security designers defined approaches that provide high level security assurance, such as encrypting data before outsourcing them to Cloud servers. Such traditional approaches bring however the disadvantage of prohibiting useful information release. This raises then the need to come up with new models and approaches for defining and enforcing security and utility policies on outsourced data. This thesis aims to address this trade-oD, while considering two kind of security policies. In the first hand, we focus on confidentiality policies specification and enforcement, which requires enforcing the secrecy of outsourced data stored by an untrusted Cloud service provider, while providing an eDcient use (e.g., searching and computing) of the outsourced data by diDerent authorized users. On the other hand, we address the problem of heterogeneous security policies (e.g., confidentiality requirements, privacy requirements, ownership requirements, etc) specification and deployment.

Keywords : Outsourced Data, Security Policy, Policy Specification, Policy Enforcement, Formal Methods

TELECOM Bretagne

n° d'ordre : 2015telb0376

Télécom Bretagne

Technopôle Brest-Iroise - CS 83818 - 29238 Brest Cedex 3

Tél : + 33(0) 29 00 11 11 - Fax : + 33(0) 29 00 10 00