Data Privacy in P2P Systems

Mohamed Jawad

UNIVERSITE DE NANTES
FACULTE DES SCIENCES ET DES TECHNIQUES

ÉCOLE DOCTORALE STIM

« SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DES MATHÉMATIQUES »

Année 2011

N° attribué par la bibliothèque

Data Privacy in P2P Systems

THÉSE DE DOCTORAT

Discipline : Informatique

Spécialité : Bases de Données

Présentée
et soutenue publiquement par

Mohamed JAWAD

Le 29 juin 2011 à l'UFR Sciences et Techniques, Université de Nantes,
devant le jury ci-dessous

Président	:	Esther PACITTI	Université de Montpellier
Rapporteurs	:	Philippe PUCHERAL, Professeur	Université de Versailles St-Quentin
		Claudia RONCANCIO, Professeur	Institut Polytechnique de Grenoble
Examinateurs	:	Esther PACITTI, Professeur	Université de Montpellier
		Patrick VALDURIEZ, Directeur de Recherche	INRIA
		Patricia SERRANO-ALVARADO, Maître de Conférences	Université de Nantes

Directeur de thèse : Patrick VALDURIEZ

Co-encadrant : Patricia SERRANO-ALVARADO

Laboratoire : **L**ABORATOIRE D'**I**NFORMATIQUE DE **N**ANTES **A**TLANTIQUE.
UMR CNRS 6241. 2, rue de la Houssinière, BP 92 208 – 44 322 Nantes, CEDEX 3.

N° ED 503-134

DATA PRIVACY IN P2P SYSTEMS

Confidentialité des Données dans les Systèmes P2P

Mohamed Jawad

⋈

favet neptunus eunti

Université de Nantes

Mohamed JAWAD

Data Privacy in P2P Systems

??+?? p.

Acknowledgements

First I would like to thank Pr. Philippe Pucheral, Pr. Claudia Roncancio and Pr. Esther Pacitti for their precious reviews on this thesis.

I am grateful to my supervisor Pr. Patrick Valduriez whose guidance and advice made me find light through the hard journey. I am heartily thankful to Dr. Patricia Serrano-Alvarado, whose encouragement, guidance and support from the initial to the final stage enabled me to develop an understanding of the subject.

I owe my deepest gratitude to Stephane Drapeau for his help on developing our service prototype, to Boris Lucas for his help on testing and validating our service and to Marco Biazzini for his genuine ideas.

Many, many thanks to my colleagues at GDD, Pascal, Philippe, Sylvie, Hala, Yann, Reza, Eduardo, Manal and the others. I cherish our insightful discussions and their essential remarks.

I am indebted to my many colleagues and friends (you know yourselves) who offered me an enjoyable atmosphere, especially Anthony with whom I shared my office and my apartment, and Mounir with whom I shared my journey.

It is a pleasure to thank those who made this thesis bearable, all my lebanese friends and family, whose presence beside me gave me the strength I need to accomplish this thesis. I will never forget Haidar, Mahmoud, Farah, Tania, Charbel, Rabab, Khalil, Ali, Fadi, and many others who have made available their support in a number of ways.

My accomplishments would not have been possible without the help of Sawsan Chhoury. I dedicate this work to her and to the soul of her father, Abou Mohsen, may he rest in peace, for he always believed in my potentials.

Finally, to my mother Hala, my father Radwan, my brother Ibrahim, and my sister Sawsan, your faith in me and your love made me what I am today, I owe this achievement and my life to you.

Data Privacy in P2P Systems

Mohamed Jawad

Abstract

Online peer-to-peer (P2P) communities such as professional ones (e.g., medical or research communities) are becoming popular due to increasing needs on data sharing. P2P environments offer valuable characteristics but limited guarantees when sharing sensitive data. They can be considered as hostile because data can be accessed by everyone (by potentially malicious peers) and used for everything (e.g., for marketing or for activities against the owner's preferences or ethics). This thesis proposes a privacy service that allows sharing sensitive data in P2P systems while protecting their privacy. The first contribution consists on analyzing existing techniques for data privacy in P2P architectures. The second contribution is a privacy model for P2P systems named PriMod which allows data owners to specify their privacy preferences in privacy policies and to associate them with their data. The third contribution is the development of PriServ, a privacy service located on top of DHT-based P2P systems which implements PriMod to prevent data privacy violations. Among others, PriServ uses trust techniques to predict peers behavior.

Keywords: Data Privacy, Purpose-based Access Control, Trust, P2P systems, distributed hash tables, Hippocratic databases.

Confidentialité des Données dans les Systèmes P2P

Résumé

Les communautés en ligne pair-a-pair (P2P), comme les communautés professionnelles (p. ex., médicales ou de recherche) deviennent de plus en plus populaires a cause de l'augmentation des besoins du partage de données. Alors que les environnements P2P offrent des caractéristiques intéressantes (p. ex., passage a l'échelle, disponibilité, dynamicité), leurs garanties en termes de protection des données sensibles sont limitées. Ils peuvent être considérés comme hostiles car les données publiées peuvent être consultées par tous les pairs (potentiellement malicieux) et utilisées pour tout (p. ex., pour le commerce illicite ou tout simplement pour des activités contre les préférences personnelles ou éthiques du propriétaire des données). Cette thèse propose un service qui permet le partage de données sensibles dans les systèmes P2P, tout en assurant leur confidentialité. La première contribution est l'analyse des techniques existant pour la confidentialité de données dans les architectures P2P. La deuxième contribution est un modèle de confidentialité, nomme PriMod, qui permet aux propriétaires de données de spécifier leurs préférences de confidentialité dans de politiques de confidentialité et d'attacher ces politiques a leurs données sensibles. La troisième contribution est le développement de PriServ, un service de confidentialité, base sur une DHT qui met en oeuvre PriMod afin de prévenir la violation de la confidentialité de données. Entre autres, PriServ utilise de techniques de confiance pour prédire le comportement des pairs.

Mots-clés : Confidentialité de données, Objectif d'Accès, Confiance, Système Pair-à-Pair, DHT, Base de Données Hippocratiques.

Discipline : Informatique
Spécialité : Bases de Données

Laboratoire : Laboratoire d'Informatique de Nantes Atlantique.
umr cnrs 6241. 2, rue de la Houssinière, bp 92 208 – 44 322 Nantes, Cedex 3.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Privacy is a fundamental human right. It underpins human dignity and other values such as freedom of association and freedom of speech. It has become one of the most important human rights of the modern age [Rot00].

We start this manuscript by showing the value of data privacy through history and the relation between data privacy and public laws. Then, we motivate this work by showing the privacy problems caused by data sharing in distributed environments. Finally, we present the thesis goal, its contributions and the organization of this manuscript.

1.1 Privacy through history

The recognition of privacy is deeply rooted in history. There is recognition of privacy in many religions, civilizations and well known philosophical discussions. For instance, in ancient Greece, Aristotle has made distinction between the public sphere of political activity and the private sphere associated with family and domestic life. Discussions about privacy, in history, were always concerned about protecting private information on personal lives from disclosure in public societies.

Afterward, discussions about privacy protections emerged in western countries. In 1361, the Justices of the Peace Act in England provided for the arrest of people who try to spy other people personal lives [Mic61]. In 1765, British Lord Camden, striking down a warrant to enter a house and seize papers wrote, "We can safely say there is no law in this country to justify the defendants in what they have done; if there was, it would destroy all

the comforts of society, for papers are often the dearest property any man can have" [vC75]. In the 1890s, the future president of United States Supreme Court Justice Louis Brandeis articulated a concept of privacy that urged that the individual has all the right "to be left alone" [WB90]. Brandeis argued that privacy was the most cherished of freedoms in a democracy, and he was concerned that it should be reflected in the Constitution. Edward Bloustein said that privacy is an interest of the human personality, it protects the inviolate personality, the individual's independence, dignity and integrity [Blo64].

In our modern days, interest in the right of information privacy increased with the advent of information technology. The surveillance potential of powerful computer systems prompted demands for specific rules governing the collection and handling of personal information. The genesis of modern legislation in this area can be traced to the first data protection law in the world enacted in the Land of Hesse in Germany in 1970. This was followed by national laws in Sweden (1973), the United States (1974), Germany (1977), and France (1978).

1.2 Data privacy and law

In the literature, there is no consensus on a definition of data privacy. It can vary depending on the domain in which it is applied. In business, data privacy is defined, in the dictionary of accounting terms [DAT10], as a set of security measures and devices employed by the accountant to assure that confidential information (e.g., client files) are not improperly accessed. In information technology, Roger Clarke[1] defines data privacy as the right of individuals to claim that data about themselves should not be automatically available to other individuals and organizations, and that, even if data is possessed by another party, the individual must be able to exercise a substantial degree of control over that data and its use.

Alain Westin [Wes67] defines data privacy as:

> *"The right of individuals to determine for themselves* ***when****,* ***how*** *and* ***to what*** *extent information about them is communicated to others."*

In this work, we adopt this definition for its clarity and pertinence. From this definition, many questions arise such as:

- Who stores the data?
- Who controls the data access?
- Who access the data?
- How data are used?

[1]Consultant specialized in strategic and policy aspects of eBusiness, information infrastructure, and data surveillance and privacy. http://www.rogerclarke.com/

- For how long should data be stored and used?
- How data owners express their privacy preferences?
- How should access rights be obtained?
- Who can (or should) claim ownership over the data collected?

Data privacy is treated by many organizations and legislations which have defined well accepted principles. In the following, we present three legislations which give a general idea on the relationship between data privacy and law. Many other from around the world are listed in [LAW].

OECD guidelines

The Organization for Economic Co-operation and Development (OECD), is one of the world's largest and most reliable source of comparable statistics, on economic and social data[2]. The OECD guidelines presented in [OEC02] state that:

- There should be limits to the *collection of personal data* and any such data should be obtained by lawful and fair means and, where appropriate, with the *knowledge or consent* of the *data owner* (Paragraph 7).
- Personal data should be relevant to the *purposes* for which they are to be used, and, to the extent necessary for those purposes, should be accurate, complete and kept up-to-date (Paragraph 8).
- The purposes for which personal data are collected should be specified not later than at the *time of data collection* and the subsequent *use limited* to the fulfillment of those purposes or such others as are not incompatible with those purposes and as are specified on each occasion of change of purpose (Paragraph 9).
- Personal data should not be disclosed, made available or otherwise used for purposes other than those specified in accordance with Paragraph 9 except: a) with the consent of the data subject; or b) by the authority of law (Paragraph 10).

The OECD guidelines state other principles such as security safeguards, openness, individual participation and accountability.

Canadian privacy act

The Canadian privacy act[3] sets out rules for how institutions of the federal government must deal with personal information of individuals. Among these rules, we cite:

[2] http://www.oecd.org/.

[3] http://laws.justice.gc.ca/en/P-21/255104.html

- A government institution may not *collect* personal information unless it relates directly to an operating program or activity of the institution (Section 4).
- When a government institution collects an individual's personal information from the individual, it must inform the individual of the *purpose* for which the information is being collected (Section 5(2)).
- Personal information under the control of a government institution may be used only for the purpose for which the information was obtained or for a *use consistent with that purpose*, unless the individual consents (Section 7).
- Personal information under the control of a government institution may not be disclosed, unless the individual consents (Section 8).

Data protection act of the French CNIL

The data protection act of the French "Commission Nationale de l'Informatique et des Libertés" (CNIL)[4] states among other principles that:

- All fraudulent, unfair or illegal *collection* of data are prohibited (Article 226-18).
- Computer processing must be done according to an *explicit end purpose* (Articles 226-21).
- It is with regard to this end purpose that one can appreciate the relevant, adequate and non-excessive nature of the data recorded, the categories of persons or organizations who may receive these data, and the *duration* for which the collected data may be stored (Article 226-20).
- The persons whose personal data are collected must be *informed* of the compulsory or optional nature of the responses, the consequences of failing to give an answer, the categories of persons or organizations who could eventually have knowledge of the data, the place where the right of access and rectification may be exercised (Article 2 of the decree of December 23^{rd} 1981).
- Any information which shows, directly or indirectly, racial origins, political, philosophical or religious opinions, trade union membership, or moral principles of the person can only be collected and recorded with the express (written) *agreement* of the person concerned (Article 226-19).

In general, all these acts underline that data privacy should consider: collection limitation, purpose specification, use limitation, data quality, security safeguards, openness, individual participation, and accountability. All these principles should be respected in order to protect data privacy. We will see in the following that, even if these laws protect data privacy from illegal actions of institutions and companies, a simple web user is able to violate data privacy due to data disclosure and sharing.

[4]http://www.cnil.fr/

1.3 Privacy and data disclosure

The disclosure of personal data by large companies has made privacy protection a critical need. Disclosure of personal data, even by mistake, can have undesirable consequences for data owners.

In January 2000, a company that sells health products online, has revealed on its website the name of its customers, their phone numbers and even information about their bank accounts and credit cards which caused a great panic among costumers.

In another example, for pharmaceutical reasons, a medical service has communicated its database on the Internet. It included the names of 4.3 million people with allergies, 923 thousand with urinary problems, and 380 thousand patients who suffer from depression. This has had serious effects on people wanting to keep their personal data confidential.

In September 2003, following the rape accusations against the basketball player Kobe Bryant, the alleged victim's medical records were subpoenaed by Bryant's defense lawyers from a Colorado hospital. After a hospital employee released the records to a judge, attorneys for the hospital asked the judge to throw out the subpoenas and destroy the records already received by him, citing state and federal medical privacy laws. Attorneys for the victim are also attempting to prevent Bryant's defense team from gaining access to her medical records from two other hospitals. However, a number of news stories have published sensitive medical information that reporters allege that they came from hospital employees. In this case, many people found their private medical information published to the public interested in the Bryant case.

These examples and many others [TRU] give us an idea about the consequences of privacy breach due to data disclosure.

1.4 Privacy in online sharing communities

In the last fifteen years, the growth of the global Internet has facilitated the rapid emergence of online interactions of dispersed groups of people with shared interests. These online groups exhibit a wide range of characteristics and serve a variety of purposes. They have varied from small groups engaged in tightly focused discussions of specific topics, to complex created worlds with hundreds of simultaneous participants. Now, they may contain millions of users linked by an interest in markets or exchange networks for goods and information [WP02].

Online communities such as professional communities (e.g., medical [MED, SER] or research communities [DOC]) are becoming popular due to increasing needs on data sharing. They often provide significant support for communities members. Yet, in order to take advantage of that support, users must frequently disclose sensitive information such as medical profiles, new research results, etc.

Most threats to privacy within an online community can be classified into two distinct groups based on the subject of the disclosure: (1) community members posting private information about someone else and (2) community members posting personal information about themselves that is then disseminated outside of the community by another member

without authorization. What happens when other community members fail to keep the potentially harmful information confidential? Traditional remedies will likely fail to protect people when members of a data sharing community violate the confidentiality of other members. In this context, the notion of *trust* between members could play a great role to handle users behavior and prevent such privacy violation.

1.5 Privacy and P2P systems

Peer-to-peer systems (P2P) became an integrated part of the Internet fabric attracting millions of users. According to recent evaluations [SW05], P2P traffic now exceeds Web traffic which was the dominant traffic on the Internet. The most popular P2P applications remain file sharing and content distribution in online communities.

Within just a few years, the huge popularity of P2P systems and the explosion of P2P research have created a large body of knowledge. P2P environments for data-centered applications offer valuable characteristics (e.g., *scalability, distribution, autonomy*) but limited guarantees concerning data privacy. They can be considered as hostile because data, that can be sensitive or confidential, can be accessed by everyone (by potentially untrusted peers) and used for everything (e.g., for marketing, profiling, fraudulence or for activities against the owner's preferences or ethics).

A study [GK03] has found a dozen examples in Kazaa [KAZ], where sensitive information like tax bills and personal e-mails are shared carelessly in most cases, but maliciously in other cases. Kazaa exposes financial information, personal files and private correspondence to millions of users around the world. These data could be used, for example, to commit fraud (especially in the financial world), to threat personal lives, or even make identity theft.

In a more recent study [Vij10], professor Eric Johnson of Dartmouth College described how university researchers discovered thousands of documents containing sensitive patient information on popular P2P networks. One of the 3,000 files discovered by the researchers was a spreadsheet containing insurance details, personally identifying information, names and diagnosis codes on more than 28,000 individuals. Another document contained similar data on more than 7,000 individuals. Many of the documents contained sensitive patient communications, treatment data, medical diagnoses and psychiatric evaluations. At least five files contained enough information to be classified as a major breach under current health-care breach notification rules.

Several P2P systems propose mechanisms to ensure some kind of privacy such as OceanStore [KBC+00], Past [RH01], Freenet [CMH+02], etc. However, these works remain insufficient. The data privacy laws have accentuated the respect of users privacy preferences. Thus P2P systems must take into account privacy preferences which is not the case in current P2P systems.

Managing data sharing, with trustworthy peers, for specific purposes and operations, is not possible without adding new services. Consider, as an applicative example, a collaborative medical research focusing on the evolution of cardiovascular diseases (e.g.,

heart attacks, atherosclerosis, etc.) depending on patient characteristics (e.g., age, diet, smoking, sports, gender, etc.). The participants of this research are scientists, doctors, patients, pharmacists, nurses, medical students, etc. In order to control disclosure on sensitive data (e.g., medical records owned by doctors or research results owned by scientists) without violating privacy, data access should respect the privacy preferences of data owners. They can be defined in the following manner:

- A doctor may allow *reading* access on her medical records to a *patient* for *seeing her diagnosis*.
- A doctor may allow *reading* access on information such as age but do not to information such as name or social security number, to *scientists* for *researching* on the evolution of the cardiovascular disease.
- A doctor may allow *writing* access on her information to *researchers* for *adding comments* on her diagnosis.
- A researcher may allow *reading* access on her research results to *doctors* for *diagnosing*.

In this P2P application, sharing data based on data owners' privacy preferences is a challenge. Besides ensuring that data disclosure should be done only to specified participants, purposes (e.g., seeing one's diagnosis, researching, etc.) and specified operations (e.g., writing, reading) defined by data owners, should be respected by data requesters. In addition, data should not be shared indistinctly with all participants (all scientists or all doctors in the system). It is necessary to consider the concept of trust among participants. For instance, doctors need to trust a researcher to share their private data with him. In this context, an efficient P2P purpose-based privacy service with trust control is needed.

1.6 Thesis goal

The goal of this thesis is the following:

- To propose privacy preserving and privacy control mechanisms which can compel P2P users to respect privacy laws and legislations.
- To contribute to the development of a novel and efficient privacy service for P2P systems. This privacy service should provide a safe environment for data sharing and storage in P2P systems while preserving data privacy.
- To promote the use of P2P systems among professional online communities and applications by enforcing P2P systems to prevent data privacy violation.

To accomplish our goal, we study interesting methods and techniques for ensuring data privacy. We also review many works that use these techniques and analyze them in order to resume the privacy guarantees needed and propose a privacy model and a privacy service to respond to these needs.

1.7 Contributions

As part of the APPA (Atlas Peer-to-Peer Architecture) project [AMPV06], the contributions of this thesis are novel techniques for data privacy protection. To be precise, our contributions in this thesis are the following.

- First, we give a state of the art that contains most recent techniques which have been proposed to protect data privacy. We give an overview of existing techniques such as access control, anonymity, trust and cryptography.

- Second, we survey works proposing data privacy protection in P2P systems. We show and compare these services on the basis of several criteria. We see for each proposal which privacy techniques are used to protect data. Then we show what sort of privacy protection these systems guarantee.

- Third, we propose PriMod, a P2P data privacy model. The goal of PriMod is to provide the basis of a framework that facilitates the prevention of data owner's privacy violation by 1) allowing them to specify their privacy preferences and by 2) restraining data requesters to make requests that specify the purpose and the operation for which data are requested.

- Forth, we propose PriServ, a privacy service on top of Distributed Hash Tables (DHT) which, based on PriMod, prevents privacy violation by limiting malicious data access. For that, we use purpose and operation based access control and trust techniques. To our knowledge, PriServ is the first proposition that introduces data access based on purposes in P2P systems.

- Fifth, the performance of our approach showed through simulation have motivated us to implement the PriServ prototype. We propose a demonstration of this prototype through a medical privacy-preserving application.

1.8 Thesis organization

The rest of this thesis is organized as follows.

- Chapter 2 presents an overview of existing privacy protection techniques. The goal is to analyze the advantages and disadvantages of these techniques used for privacy protection in order to choose which techniques are more suitable for our work.

- Chapter 3 offers a review and comparison of several P2P services that use privacy techniques to protect data privacy. The comparison is based on (a) which privacy techniques are used by these systems and (b) what privacy guarantees do these systems offer.
- Chapter 4 introduces PriMod, a privacy model for P2P systems that takes into account relevant techniques for privacy protection.
- Chapter 5 presents PriServ, a privacy service for DHTs, which offer privacy protection in DHT-based P2P systems without introducing a great communication overhead.
- Chapter 6 gives an experimental validation of PriServ through a cost analysis, simulation and a prototype implementation. The PriServ prototype is demonstrated through a medical privacy-preserving application.

Finally, we conclude and highlight future directions of research.

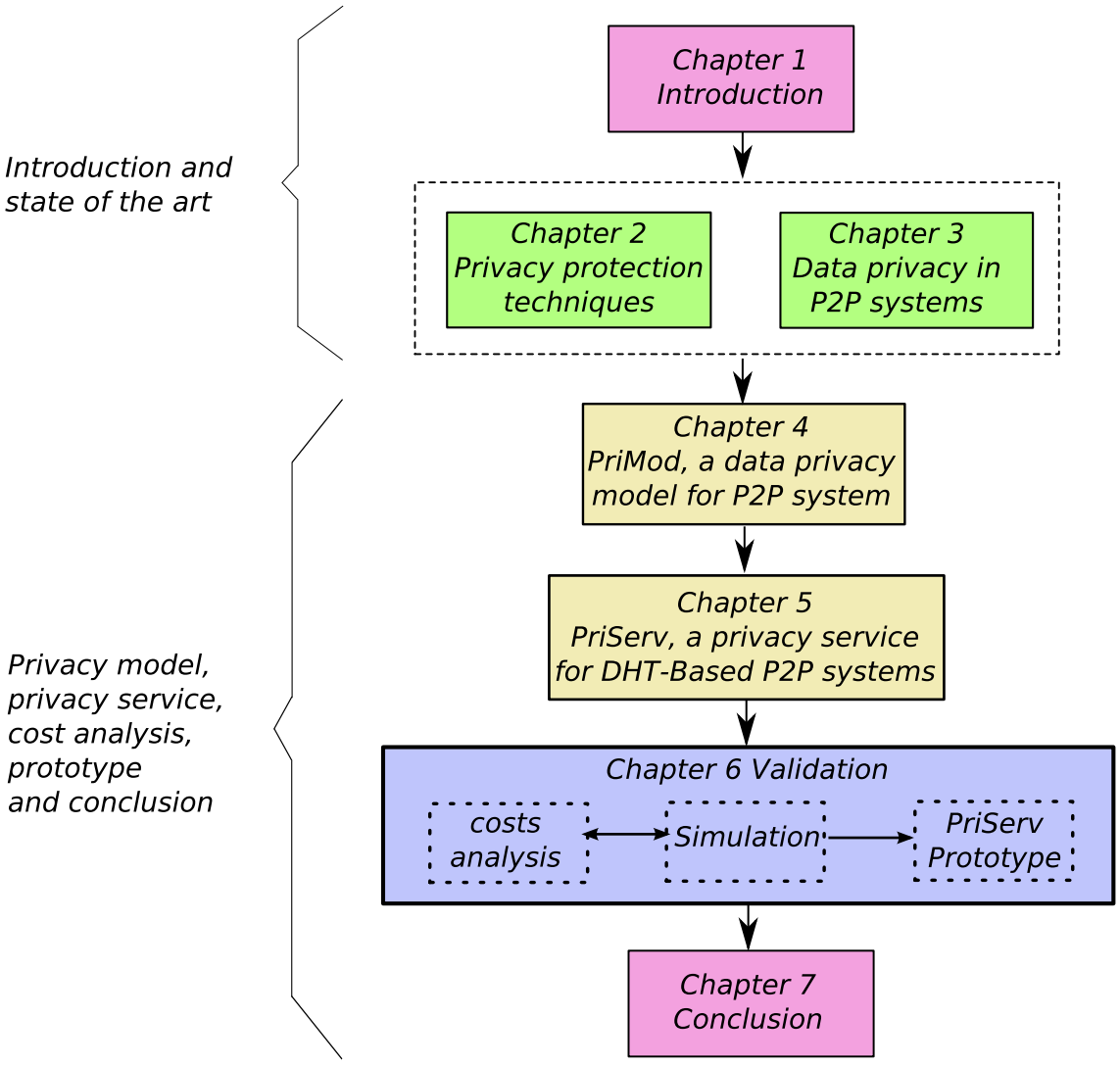


Figure 1.1: Thesis organization

List of Publications

International Conferences

- Mohamed Jawad, Patricia Serrano Alvarado and Patrick Valduriez. Protecting Data Privacy in Structured P2P Networks. In *Proceedings of the Second International Conference on Data Management in Grid and P2P Systems (Globe)*, August 31-September 4, 2009, Linz, Austria.

- Mohamed Jawad, Patricia Serrano Alvarado, Patrick Valduriez and Stéphane Drapeau. A Data Privacy Service for Structured P2P Systems. In *Proceedings of the Mexican International Conference in Computer Science (ENC)*, September 21-25, 2009, Mexico City, Mexico.

International Workshops

- Mohamed Jawad, Patricia Serrano Alvarado and Patrick Valduriez. Design of PriServ, a Privacy Service for DHTs. In *Proceedings of the 2008 international workshop on Privacy and Anonymity in Information Society (PAIS)*, March 29-29, 2008, Nantes, France.

National Conferences

- Mohamed Jawad, Patricia Serrano Alvarado, Patrick Valduriez and Stéphane Drapeau. Data Privacy in Structured P2P systems with PriServ. *In actes des 25èmes journées Bases de Données Avancées (BDA)*, October 20-23, 2009, Namur, Belgium.

- Mohamed Jawad, Patricia Serrano Alvarado, Patrick Valduriez and Stéphane Drapeau. Privacy Support for Sensitive Data Sharing in P2P Systems. *In actes des 27èmes journées Bases de Données Avancées (BDA)*, October 24-27, 2011, Rabat, Morroco.

Chapter 2

Privacy protection techniques

In order to define the problem addressed in this thesis, this chapter provides a state of the art of privacy protection techniques. In particular, we present access control techniques, Hippocratic databases, anonymity, trust and cryptography techniques.

2.1 Introduction

The end of the twentieth century and early years of the twenty first century saw rapid advancements in telecommunications, hardware and software. The rapid growth and widespread use of data processing, conducted through the Internet, fueled the need for better methods of protecting computers and the information stored, processed and transmitted. Academic disciplines of computer and information security emerged along with numerous professional organizations, all sharing the common goals of ensuring the security and reliability of information systems. The core principles of information security are: privacy, integrity and availability. In this thesis, we focus on data privacy.

Recent research has been concentrated on ameliorating ancient privacy techniques and proposing novel algorithms to preserve privacy in information systems. Among these, we list:

- Access control techniques, which allow to limit unauthorized access to private data.
- Cryptography techniques, which are the first techniques used to provide data security and privacy, and remain currently the most used ones.

- Anonymity techniques, which protect privacy by making users indistinguishable of other users and hardly identified.
- Trust techniques, which add more privacy protection in information systems by handling the trustworthiness of users.
- Hippocratic databases, which integrate a new notion to privacy protection, the purpose-based access control.

In this chapter, we give an overview of these five types of privacy techniques which have been largely used in the recent decade and have been proved to be very efficient in resolving some of the privacy greatest concerns in information systems.

The rest of the chapter is organized as follows. Section 2.2 presents access control techniques. Section 2.3 introduces Hippocratic databases. Section 2.4 shows anonymity techniques. Section 2.5 presents trust techniques. Section 2.6 shows cryptography techniques. Then, Section 2.7 concludes.

2.2 Access control

Access control techniques are used to delimitate the rights of individuals or application programs on objects [FED].

In any access control model, the entities that can perform actions in the system are called *subjects*, and the entities representing resources to which access may need to be controlled are called *objects* [Hof77]. Each subject has access to certain objects. This access information may be stored by subject or by object. In the former case, the access permissions are called users capabilities. If the permissions are stored by objects, they are called access control lists (ACLs). Depending on which type of permission is used, access control models tend to fall into one of two classes: those based on capabilities and those based on ACLs.

In a capability-based model, holding an unforgeable reference or capability to an object provides access to the object. This can be analogous to how possession of a house key grants access to a house. Access is conveyed to another subject by transmitting such a capability over a secure channel.

In an ACL-based model, a subject's access to an object depends on whether its identity is on a list associated with the object. This can be analogous to how a bouncer at a private party would check your ID to see if your name is on the guest list. Access is conveyed by editing the list. Different ACL systems have a variety of different conventions regarding who or what is responsible for editing the list and how it is edited.

Both, capability and ACL based models, have mechanisms to allow access rights to be granted to all members or a group of subjects. Access control systems provide the essential services of identification, authorization, and accountability where:

- Identification determines who can log in a system, and the association of users with the subjects that they are able to control as a result of logging in.

- Authorization determines what a subject can do.
- Accountability determines what a subject (or all subjects associated with a user) did.

In the next, we present identification (cf., Section 2.2.1), authorization (cf., Section 2.2.2) and accountability (cf., Section 2.2.3). Then, we introduce the most used access control techniques (cf., Section 2.2.4).

2.2.1 Identification

Identification is the process of verifying that an identity is bound to the user that makes an assertion or claim of identity [Ale10]. The identification process assumes that there was an initial validation of the identity. The only requirements for the user identifier is that it must be unique within its security domain. Identifiers are commonly based on the following factors:

- A secret known only by authorized users such passwords or personal identification numbers (PIN).
- A physical material held by authorized users such as smart cards or security tokens.
- The location of the user for example inside or outside a company firewall.
- A real characteristic of the user such as her fingerprint or her voice characteristics.

2.2.2 Authorization

Authorization determines what a subject can do on the system [Ale10]. Most modern systems define sets of permissions that are variations or extensions of three basic types of access:

- Read (R): The subject can read data content.
- Write (W): The subject can change the contents of a data with the following tasks: Create, Update, Delete.
- Disclose (D): The subject can share the data with other users she knows.

These rights and permissions are implemented differently depending on the access control technique (cf., Section 2.2.4).

2.2.3 Accountability

Accountability uses audit trails (records) and logs to associate a subject with her actions [Ale10]. Audit trails and logs are important for detecting security violations or re-creating security incidents. Many systems can generate automated reports based on certain predefined criteria or thresholds, known as clipping levels. For example, a clipping level may be set to generate a report for the following:

- More than three failed logon attempts in a given period.
- Any attempt to use a disabled user account.

These reports help a system administrator or security administrator to more easily identify possible break-in attempts.

2.2.4 Access control techniques

The three most widely recognized and used access control models are Discretionary Access Control (DAC), Mandatory Access Control (MAC) and Role-Based Access Control (RBAC). Recently, with the introduction of the notion of *purpose* by many legislations, a Purpose-Based Access Control (PBAC) has been proposed to add more semantics to access control models. In the following we expose the main characteristics of each of these access control models.

2.2.4.1 Discretionary access control

DAC is based on access policies determined by the owner of an object [Kar85]. The owner decides who are allowed to access the object and what privileges they have. Two important concepts in DAC are:

- Data ownership. Every object in the system has an owner. In most DAC systems, each object's initial owner is the subject that creates it. The access policy for an object is determined by its owner.
- Access permissions. These are the privileges that an owner can assign to other subjects on specific resources.

Commercial operating systems like UNIX and Windows uses DAC to allow owners to control the access to their objects.

2.2.4.2 Mandatory access control

MAC is based on access policies determined by the system, not by the owner [BL73]. It is used in multilevel systems that process highly sensitive data, such as classified government and military information. Two important concepts in MAC are:

- Sensitivity levels. In a MAC-based system, all subjects and objects must have levels assigned to them. A subject's sensitivity level specifies its level of trust in the system. An object's sensitivity level specifies the level of trust required for accessing this object. In order to access a given object, the subject must have a sensitivity level equal to or higher than the requested object sensitivity level.

- Data disclosure. Controlling the disclosure of information for other systems is a critical function of MAC-based systems, which must ensure that sensitivity levels are properly maintained anywhere and anytime so that sensitive information is appropriately protected.

Many systems implement the MAC model such as the NSA research project SELinux [SEL], FreeBSD [FRE] and Oracle Label Security [OLS].

2.2.4.3 Role-based access control

RBAC [San96,SFK00] is based on access policies determined by the system administrator. It is used in commercial applications and also in military systems, where multilevel security requirements may also exist. RBAC is non-discretionary, however it can be distinguished from MAC primarily in the way permissions are handled. While MAC permissions are based on users' sensitivity levels, RBAC permissions are based on users' roles. Each user is assigned a set of roles and a set of permissions is assigned to each role. Three important concepts in RBAC are:

- Role assignment. A subject can execute an operation only if the subject has selected or been assigned a role.

- Role authorization. Subjects can take only roles for which they are authorized.

- Operation authorization. A subject can execute an operation only if the operation is authorized for the subject's active role. This guarantees that subjects can execute only operations for which they are authorized.

Additional constraints may be applied as well, and roles can be combined in a hierarchy where higher-level roles subsume permissions owned by sub-roles.

Systems including Microsoft Active Directory [ACT], Microsoft SQL Server [SQL], Solaris [SOL] and many others effectively implements RBAC.

2.2.4.4 Purpose-based access control

PBAC [BL08] is a new model proposed to add more semantics to access control. In PBAC, purpose information is associated with data. A purpose describes the reason(s) for data collection and data access. In this model, the notion of purpose plays a central role as the purpose is the basic concept on which access decisions are made. Access is granted based on the intented purposes of the subjects. Each subject is required to state her access

purpose when trying to access an object. The system validates the stated access purpose by the user. Three important concepts in PBAC are:

- Intended purposes. An intended purpose is the purpose associated with data and thus regulating data accesses. Intended purposes can be viewed as brief summaries of privacy policies for data, stating for which purposes data can be accessed.
- Access purposes. An access purpose is the purpose of a particular data access, which is determined or validated by the system when the data access is requested.
- Access purpose compliance. An access decision is made based on the relationship between the access purpose and the intended purpose of data. That is, an access is allowed only if the access purpose is included in the context of use implied by the intended purpose. In this case, the access purpose is "compliant" with the intended purpose. An access is denied if the implication of the intended purpose does not include the access purpose.

Since privacy laws are concerned with the purposes for which data objects are used, traditional access control models cannot easily achieve privacy protection, and the notion of purpose should play a major role in access control to protect privacy. PBAC has been included in many privacy-preserving schemes. The following section presents Hippocratic databases [AKSX02] which introduce the notion of purpose in relational databases.

2.3 Hippocratic databases

> *"And about whatever I may see or hear in treatment, or even without treatment, in the life of human beings - things that should not ever be blurted out outside - I will remain silent, holding such things to be unutterable" - Hippocratic Oath*

Inspired by the Hippocratic oath and its tenet of preserving privacy, Hippocratic databases aim to incorporate privacy protection within relational database systems [AKSX02]. Hippocratic databases have been proposed to design database systems that include purpose-based data privacy. The important concepts in Hippocratic databases are:

- Privacy policies. A privacy policy defines for each attribute of a table (a) the usage purpose(s), (b) the external-recipients and (c) the retention period.
- Privacy authorizations. A privacy authorization defines which purposes each user is authorized to use on which data.

In a Hippocratic database, queries are submitted along with their intended purpose. Query execution preserves privacy using query modification and restrictions by column/row/cell.

2.3.1 Founding principles of Hippocratic databases

The ten founding principles of Hippocratic databases articulate what it means for a database system, to responsibly manage private information. They also define what a private information donor can expect from a database system that claims to be Hippocratic.

1. Purpose Specification. For personal information stored in the database, the purposes, for which the information has been collected, shall be associated with that information.

2. Consent. The purposes associated with personal information shall have the consent of the donor of the personal information.

3. Limited Collection. The personal information collected shall be limited to the minimum necessary for accomplishing the specified purposes.

4. Limited Use. The database shall run only those queries that are consistent with the purposes for which the information has been collected.

5. Limited Disclosure. The personal information stored in the database shall not be communicated outside the database for purposes other than those for which there is a consent of the donor of the information.

6. Limited Retention. Personal information shall be retained only as long as necessary for the fulfillment of the purposes for which it has been collected.

7. Accuracy. Personal information stored in the database shall be accurate and up-to-date.

8. Safety. Personal information shall be protected by security safeguards against theft and other misappropriations.

9. Openness. A donor shall be able to access all information about her stored in the database.

10. Compliance. A donor shall be able to verify compliance with the above principles. Similarly, the database shall be able to address a challenge concerning compliance.

In the following, we concentrate on three Hippocratic approaches [AKSX02, LAE+04, ABF+04] which are very relevant to this thesis: purpose specification, limited disclosure, and auditing compliance.

2.3.1.1 Purpose specification

Purpose specification is the cornerstone of a Hippocratic database. It states that purposes should be specified and attached to data items to control their usage. Hippocratic databases have been envisioned being used in a wide variety of rich environments (e.g., finance, insurance, and health care). [AKSX02] states that developing a policy specification language for these rich environments that strikes a good balance between expressibility and usability is a difficult problem. Simple specification language such as P3P [CLM+02] can be used as a starting point.

Platform for Privacy Preferences (P3P), developed by the World Wide Web Consortium, is an emerging standard. Its goal is to enable users to gain more control over the use of their personal information on web sites they visit. P3P provides a way for a Web site to encode its data-collection practices in a machine-readable XML format known as a P3P policy [CLM+02], which can be programmatically compared against a user's privacy preferences [Lan01]. A major criticism of P3P has been that while P3P provides a technical mechanism for ensuring that users can be informed about privacy policies before they release personal information, it does not provide a mechanism for making sure sites act according to their stated policies [KEFP02, Rot01].

[KSW02, KPR01] propose ideas for reducing the complexity of the policy language which include arranging purposes in a hierarchy. Subsumption relationships may also be defined for retention periods and recipients.

2.3.1.2 Limited disclosure

Limited disclosure is a vital component of a data privacy management system. It states that the private data shall not be disclosed for purposes other than those defined by the data owner. [LAE+04] proposes a scalable architecture for enforcing limited disclosure rules and conditions at the database level. For enforcing privacy policies in data disclosure, privacy policies can be stored and managed in the database. These policies are expressed in high-level privacy specification languages (e.g., P3P). Enforcing privacy policies does not require any modification to existing database applications.

Two models are presented in [LAE+04] for cell-level limited disclosure enforcement in relational databases: table semantics and query semantics. The table semantics model defines a view of each data table for each (purpose, recipient) pair, based on the associated privacy semantics. These views combine to produce a coherent relational data model for each (purpose, recipient) pair, and queries are executed against the appropriate database version. The query semantics model removes prohibited data from a query's result set based on the purpose, recipient, and the query itself. Authors provide techniques for enforcing a broad class of privacy policies by automatically modifying all queries that access the database in a way that the desired disclosure semantics is ensured. They examine several implementation issues, including privacy meta-data storage, query modification algorithms, and structures for storing conditions and individual choices.

2.3.1.3 Auditing compliance

Auditing compliance is another vital component of a Hippocratic database. Here the idea is to verify that the database adheres to its declared data privacy policies. [ABF+04] proposes a system that can be used to audit whether the database system executed a query, in the past, that accessed the specified data.

During normal operation, the text of every query processed by the database system is logged along with annotations such as the date at which the query was executed, the user submitting the query, and the query's purpose. The system uses triggers to capture and record all updates or writes in backlog tables for recovering the state of the database at any past point in time. Read queries, which are usually predominant, do not write any tuple to the backlog database.

To perform an audit, the auditor formulates an audit expression that declaratively specifies the data of interest. Audit expressions are designed to be identical to the SQL queries. This allows audits to be performed at the level of a cell of a table. The audit expression is processed by the audit query generator, which first performs a static analysis of the expression to select a subset of logged queries that could potentially disclose the specified information. It then combines and transforms the selected queries into an audit query by augmenting them with additional predicates derived from the audit expression. This audit query, expressed in standard SQL, when run against the backlog database, yields the precise set of logged queries that accessed the designated data.

Subsequent works have proposed solutions for implementing Hippocratic databases. In [ABG+05], authors address the problem of how current relational DBMS can be transformed into their privacy-preserving equivalents. From specifications of privacy policies, they propose an algorithm that defines restrictions (on columns, rows and cells) to limit data access. In [BBL05], authors propose query modification techniques and RBAC to ensure data privacy based on purposes. They propose to organize purposes in a tree hierarchy where the root is the most general purpose and the leafs the more specific ones. In this way, if data access is allowed for a purpose x, all descendant purposes of x are also allowed. They also propose data labeling (with allowed purposes) at different granularity levels (table, column, row, cell). In addition, they propose some SQL modifications to include purposes, for instance **Select** column-name **From** table-name **For** purpose-name.

Hippocratic Database is applicable to any industry with disclosure management concerns (e.g., Healthcare, Finance, Government, etc.). IBM has developed a technology set [IBM] to enforce Hippocratic databases in disclosure management systems. Currently, the IBM technology set includes:

- *Active enforcement* which automates cell-level, policy-based disclosure management such that databases only return data that is consistent with company policies, applicable legislation, and customer preferences.

- *Compliance auditing* which records all queries and changes to the database and uses this information to construct detailed audit trails that specify the user, recipient,

purpose, time, and exact (cell-level) information disclosed for any particular database query.

- *Secure Information Sharing* which allows two parties to share information about intersections between data sets without compromising the privacy or security of the remaining data.
- *Privacy-preserving data mining* which preserves privacy at the individual level, while still allowing accurate data mining models at the aggregate level.
- *Database watermarking* which allows one to detect data theft and assert ownership rights over pirated copies.
- *Order preserving encryption* which enables database systems to execute queries over encrypted data without incurring significant performance hit or unnecessary cryptographic calls and still being able to utilize the existing database functionality.

Hippocratic databases are the first privacy techniques that included the notion of purpose in relational databases. They are essential to users who would like to know for which purpose their data are used. Enforcing Hippocratic databases in distributed environments is a challenge that we address in next chapters.

2.4 Anonymity

Anonymity is best described by the root of the word itself which means simply "no name". Anonymity techniques are majorly used to make a user (respectively a datum) indistinguishable from other users (respectively data) thus providing its anonymity among a group of users (respectively data set). The result of assuming anonymity is the desire not to be connected with some event or events. It can help to promote freedom of expression with writers and journalists. It can help to protect human rights and persons reporting illegal activities, persons seeking help for problems like AIDs, harassment, racial issues, alcohol, gambling or drug abuse.

Next, we show the following anonymity techniques, which are presented in [FWCY10, PH09]: identity-based anonymity (cf., Section 2.4.1), pseudonymity (cf., Section 2.4.2), anonymous communication (cf., Section 2.4.3) and k-anonymity and similar techniques (cf., Section 2.4.4).

2.4.1 Identity-based anonymity

One way to achieve anonymity is to hide user's identity. To do this, there has to be an appropriate set of users with potentially the same attributes. Identity-based anonymity is defined as the state of not being identifiable within the set of users, which is called the anonymity set. Users may be anonymous only with their respective anonymity set, which may vary over time. Sets for each user may be disjoint, the same, or they may overlap.

As a security requirement for anonymity, the probability that a verifier can successfully determine the real user is exactly 1/n, where n is the number of users in the anonymity set.

Identity-based anonymity is almost always associated with illegal acts such as illegal downloads. But, the point of mentioning this is to establish the fact that anonymity is not illegal or immoral by itself. By using numbers instead of patient names, anonymity protects the identity, and thereby, the privacy of patients whose medical records are used in a study of medical care outcomes.

2.4.2 Pseudonymity

Other forms of anonymity are associated with pen names or pseudonyms. Pseudonyms are generally dynamic identifiers, or names of the users that are hard to be linked to the real identities without the shared secret keys. In other words, a pseudonym is an identifier of a user other than one of the users' real names.

Since the users' true identities are kept private, this can be useful for users who would like to act freely in a system without ethical differences or racialism. However on the other hand, this can be an excellent opportunity to misbehaved users to act maliciously without getting caught.

2.4.3 Anonymous communication

Another form of anonymity are anonymous communications which are largely used in distributed systems. Anonymous communications aim to preserve communication privacy within the shared public network environment. While end-to-end encryption can protect the data content from adversarial access, it does not conceal all the relevant information that two users are communicating. Adversaries can still learn significant information about the traffic carried on the network and the physical world entities. The exposure of network addresses may result in a number of severe consequences.

On a tactical military communication network, an abrupt change in the traffic pattern may indicate some forthcoming activities. This can be extremely dangerous in that adversaries can easily identify critical network nodes and then launch targeted attacks on them. This makes source privacy an essential security requirement for government and military communications.

The research on privacy preserving communications was initiated by Chaum in 1981 [Cha81]. Since then, research in anonymous communications has been extended to many areas such as routing mechanisms, broadcasting systems and P2P communication systems. Works in this domain (e.g., onion routing, mix networks, crowds, Anonymizer) generally aim to make communication ambiguous in order to make it difficult to malicious users to collect information about the network environment.

2.4.4 K-anonymity and similar techniques

The concept of k-anonymity [SS98] tries to capture, on the private table to be released, one of the main requirements followed by the users releasing the data, and according to which the released data should be indistinguishably related to no less than a particular number k of users.

The set of attributes included in the private table, also externally available and therefore exploitable for linking, is called quasi-identifier. The k-anonymity requirement states that each release of data must be such that every combination of values of quasi-identifiers can be indistinctly matched to at least k respondents.

The enforcement of k-anonymity requires the preliminary identification of the quasi-identifiers. Then generalization and suppression techniques are used for providing anonymity. Generalization consists in substituting the values of a given attribute with more general values, while suppression consists on deleting tuples in order to reduce the amount of generalization necessary to satisfy the k-anonymity constraint.

Other techniques for providing anonymity exists such as l-diversity, (X, Y)-privacy, t-closeness etc. [FWCY10] proposes a survey that systematically summarizes and evaluates these techniques. It studies the challenges in practical data publishing and proposes future research directions.

Anonymity techniques are important to protect user or data privacy, however it may facilitate illegal behavior. In the following we present trust management techniques which can be used in addition to anonymity techniques in order to handle the trustworthiness of users without revealing their true identities (i.e., users with pseudonyms could be assigned trust levels even if their identities are kept private (e.g., ebay)).

2.5 Trust management

Trust management techniques have been proposed as mechanisms that allow potentially unknown parties to decide whom is trusted enough to provide or access requested data.

Trust management techniques have been developed for this specific purpose and have received considerable interest from the research community [BFL96, SLB06, KSGM03, PSD02]. They allow unknown parties to access resources by showing appropriate credentials that prove their qualifications to propose/get data. They can be complementary to classical access control mechanisms as they allow the addition of new restrictions and conditions without the need to rewrite the services enforcing access control.

The concept of reputation is closely linked to that of trust. Reputation is often considered as a collective measure of trustworthiness based on ratings from parties in a community and can be used to establish trust relationships [DDCdVPS03]. The basic idea behind reputation management is to let remote parties rate each other and use the aggregated ratings of a given party to derive a reputation score. Reputation can then be used by other parties when deciding whether or not to work with that party in the future.

In the following, we present some basic trust and reputation concepts (cf., Section 2.5.1). Then, we show some trust management techniques (cf., Section 2.5.2).

2.5.1 Trust and reputation concepts

Trust and reputation techniques are based on many concepts. In the following we show some concepts defined in [ST04], in particular, trust levels, types of reputation, credential verification and addition of new users.

Trust levels. Trust models typically use levels to represent the trust one user has in another. These levels may either be discrete or continuous depending on the needs of the application and the type of trust model used. Some trust models employ binary values, implying that a user either completely trusts or distrusts another one. Tagging users as either completely trustworthy or untrustworthy does not allow a user to express partial trust. Continuous trust levels, on the other hand, provide more expressive power to define trust relationships.

Trust levels initialization depends on many parameters. [XL04] initializes trust levels by using these parameters:

- Feedback in terms of amount of satisfaction. The feedback a peer receives from other peers reflects how well this peer has fulfilled its part of the service agreement.
- Number of transactions. A peer may increase its trust value by increasing its transaction volume to hide the fact that it frequently misbehaves. Thus the total number of transactions a peer performs is considered for the trust levels calculation.
- Credibility of the feedback sources. The feedback from those peers with higher credibility should be weighted more than those with lower credibility.
- Transaction context. This factor is used for differing mission-critical transactions from less or non critical ones.
- Community context. This factor is used for addressing community-related characteristics and vulnerabilities.

Types of reputation. In reputation models, users may use three kinds of reputation information to determine the extent of trust in users: positive reputation, negative reputation, or a combination of both. In a positive reputation mechanism, users only exchange information about successful transactions. In a negative reputation mechanism, on the other hand, users are generally assumed to be good and reputation is expressed only as negative feedback or complaints that are distributed to other peers. Both mechanisms while useful are incomplete by themselves. For example, relying only upon successful transactions may result in users ignoring the recent malicious actions of a "trustworthy" peer. The drawback of relying only on a negative reputation-based scheme is that a user

may end up trusting a malicious peer if it does not have access to existing complaints. A combination of positive and negative reputations makes the reputation mechanism more robust and reliable.

Credential verification. In trust models that explicitly use credential verification, credential evaluation prevents malicious users from taking on the identity of other users. In one common technique every user generates a public and private key pair. Any message sent out is signed by the sender's private key and authenticated by the receiver using the sender's public key. While credential verification can be easily added onto most trust models, a lot of trust models do not explicitly specify whether their models actually employ credential verification to establish user trustworthiness.

Addition of new users. When a new user joins an existing system, it does not possess trust-based knowledge about other users. This may hold her back from interacting with other users. Similarly, existing users in the system may tend to isolate the new user since they lack trust information about her. A trust model, therefore, should have a low barrier of entry for new users so that they can easily participate in the system. Yet, at the same time, the trust model should provide sufficient measures to protect the system if the new user turns out to be malicious. Addition of new users is very effective while initializing the system.

2.5.2 Trust management techniques

According to [ST04], trust management techniques are classified into three categories: credential-based trust management, reputation-based trust management, and trust management based on social networks. This categorization is based on the approach adopted to establish and evaluate trust relationships between users.

2.5.2.1 Credential-based trust systems

In credential-based trust management systems such as [BFL96, KCFP01, Yu01, LMW02, Yao03], users use credential verification to establish a trust relationship with users. The primary goal of such systems is to enable access control. Therefore their concept of trust management is limited to verifying credentials and restricting access to resources according to application-defined policies [GS00]. A user (i.e., data owner) provides to other users (i.e., data requesters) access to restricted data only if it can verify the credentials of the requester either directly or through a web of trust. This is useful by itself only for those applications that assume implicit trust in data owners. Since these credential-based trust mechanisms do not incorporate the need of the user to establish trust in the data owner, by themselves they do not provide a complete generic trust management solution for decentralized applications.

2.5.2.2 Reputation-based systems

Reputation-based trust management systems provide a mechanism by which a data user may evaluate its trust in the reliability of the data and the server which stores them. Examples of such systems include XREP [DdVP+02], NICE [SLB06], EigenTrust [KSGM03], etc. Users in such systems establish trust relationships with other users and assign trust values to these relationships. A trust level assigned to a trust relationship is a function of the combination of the user global reputation and the evaluating user perception of that user.

Reputation-based solutions do not require any prior experience with the user for reputation to be used to infer trustworthiness. It is then suitable for establishing initial trust. Users in such systems establish trust relationships with other users and assign trust levels to these relationships.

[Ser06] provides a method to categorize peer-to-peer reputation systems. It identifies three basic components for comparison, each one has several properties:

- Information gathering. This component is responsible for collecting information on the behavior of peers, which will be used to determine how trustworthy they are. In this component, are considered, identity scheme, information sharing, sources of information, information integrity and dealing with strangers.
- Scoring and ranking. This component is responsible of computing a reputation score for the peers based on expected reliability. In this component, are considered inputs, outputs and peer selection.
- Taking action. This component is used by reputation systems to motivate peers to positively contribute to the network and/or punish adversaries who try to disrupt the system. In this component, are considered incentives and punishments.

[Ser06] gives examples of research in the area of trust and reputation. A variety of research papers and implementations are referenced to illustrate ideas and provide the reader avenues for further investigation.

2.5.2.3 Trust systems based on social networks

The third kind of trust management systems uses social relationships between users to compute trust levels. In particular, they analyze a social network which represents the relationships existing within a community, then they define conclusions about users reputation based on different aspects of the social network. Examples of such trust management systems include Regret [SS02] which defines trust groups using the social network, and NodeRanking [PSD02] which finds experts using the social network.

Trust techniques remain important in largely distributed environments. They can be used to predict the behavior of unknown users and thus to provide protection from malicious acts.

2.6 Cryptography

Cryptography is the practice and study of hiding information [GRE84]. Modern cryptography intersects the disciplines of mathematics, computer science, and engineering. Applications of cryptography include ATM cards, computer digital passwords, and electronic commerce. Cryptography referred almost exclusively to encryption, which is the process of converting ordinary information (plain text) into unintelligible cipher text. Decryption is the reverse, in other words, moving from the unintelligible cipher text back to plain text. A cipher is a pair of algorithms that create the encryption and the reversing decryption. The detailed operation of a cipher is controlled both by the algorithm and in each instance by a key. This is a secret parameter for a specific message exchange context[1]. Keys are important, as ciphers without variable keys can be trivially broken with only the knowledge of the cipher used and are therefore less than useful for most purposes [Kah74].

In the following, we present three types of cryptography: symmetric-key cryptography (cf., Section 2.6.1), asymmetric-key cryptography (cf., Section 2.6.2) and hybrid cryptography (cf., Section 2.6.3).

2.6.1 Symmetric-key cryptography

Symmetric-key cryptography refers to encryption methods in which both the sender and the receiver share the same key or, less commonly, in which their keys are different, but related in an easily computable way. This was the only kind of encryption publicly known until June 1976 [DH76]. Since then, other kinds of encryption have been proposed. The modern study of symmetric-key ciphers relates mainly to the study of block ciphers and stream ciphers and to their applications.

Block ciphers. Block ciphers take as input a block of plain text and a key, and output a block of cipher text of the same size. Since plain texts are almost always longer than a single block, some method of knitting together successive blocks is required. Several methods have been developed, some with better security than others, in one aspect or another. The Data Encryption Standard (DES) [Dav76] and the Advanced Encryption Standard (AES) [AES01] are block cipher designs which have been designated cryptography standards by the US government. Despite its deprecation as an official standard[2], DES, especially its still-approved and much more secure triple-DES variant [CJM96], remains quite popular. It is used across a wide range of applications, from ATM encryption to e-mail privacy and secure remote access. Many other block ciphers have been designed and released, with considerable variation in quality [BLO].

Stream ciphers. Stream ciphers, in contrast to block ciphers, create an arbitrarily long stream of key material, which is combined with the plain text bit-by-bit or character-by-

[1] Message exchange is the process that has most benefited from cryptography

[2] In January 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes.

character. In a stream cipher, the output stream is created based on a hidden internal state which changes as the cipher operates. That internal state is initially set up using the secret key material. RC4 [Riv92b] is a widely used stream cipher.

Cryptographic hash functions. Cryptographic hash functions are a third type of cryptographic algorithms. They are one-way functions that take a message of any length as input, and output a short, fixed length hash value which can be used in digital checksums. Good hash functions are collision-resistant (i.e., two different messages do not produce the same hash value). MD4 [Riv90] is a long-used hash function which is now broken. MD5 [Riv92a], a strengthened variant of MD4, is also widely used but also broken in practice. The U.S. National Security Agency (NSA) developed the Secure Hash Algorithm series [SHA02] of MD5-like hash functions: SHA-0 was a flawed algorithm that the agency withdrew. SHA-1 is widely deployed and more secure than MD5, but cryptanalysts have identified attacks against it. The SHA-2 family have been improved on SHA-1, but they are not yet widely deployed. NIST[3] thought it was prudent from a security perspective to develop a new standard to significantly improve the robustness of NIST's overall hash algorithm toolkit. Thus, a hash function design competition is underway and meant to select a new U.S. national standard, to be called SHA-3, by 2012 [CSR].

Symmetric-key algorithms are generally much less computationally intensive than other cryptography algorithms. For instance, symmetric key algorithms are typically hundreds to thousands times faster than asymmetric key algorithms (cf., Section 2.6.2). One disadvantage of symmetric-key algorithms is the requirement of a shared secret key, with one copy at each end. To limit the impact of a potential discovery by a cryptographic adversary, keys should be changed regularly and kept secure during distribution and in service. The process of selecting, distributing and storing keys is known as key management [RH03], and it is difficult to achieve it reliably and securely.

2.6.2 Asymmetric-key cryptography

Asymmetric-key cryptography, also known as public-key cryptography, is an approach which involves the use of asymmetric-key algorithms. Unlike symmetric key algorithms, it does not require a secure initial exchange of one or more secret keys to both sender and receiver. The asymmetric key algorithms are used to create a mathematically related key pair: a secret private key and a published public key.

Asymmetric algorithms can be used for two objectives: data encryption and digital signature schemes. In asymmetric-key encryption, the message, which has been encrypted by using the public key, can only be decrypted by using the private key. The Diffie-Hellman [DH76] and RSA [RSA78] algorithms, in addition to being the first publicly known examples of high quality asymmetric-key cryptography algorithms, have been among the most widely used.

[3]The national institute of standards and technology is the U.S. standards authority.

In addition to encryption, asymmetric-key cryptography can be used to implement signature schemes. In these schemes, there are two algorithms: one for signing, in which a private key is used to process the message, and one for verification, in which the matching public key is used with the message to check the validity of the signature. Digital signatures are reminiscent of ordinary signatures. They have the characteristic that they are easy for a user to produce, but difficult for anyone else to forge. Digital signatures can also be permanently tied to the content of the message being signed. They cannot then be moved from one document to another, for any attempt will be detectable. RSA and DSS [BP09] are two of the most popular digital signature schemes. Digital signatures are central to the operation of public key infrastructures and to many network security schemes (e.g., SSL/TLS, VPNs, etc.).

Asymmetric-key algorithms are most often based on the computational complexity of hard problems, often from theory of number. For example, the hardness of RSA is related to the integer factorization problem. Because of the difficulty of the underlying problems, most public-key algorithms involve operations such as modular multiplication and exponentiation, which are much more computationally expensive than the techniques used in most block ciphers, especially with typical key sizes. This is the main disadvantage of such systems. One of the main advantages of asymmetric-key cryptography is that it does not need secret key management because keys are public.

2.6.3 Hybrid cryptography

Asymmetric-key cryptography is convenient because it does not require the sender and the receiver to share a common secret in order to communicate securely. However, it often relies on complicated mathematical computations and it is thus generally much more inefficient than comparable symmetric-key cryptography. In many applications (e.g., real-time or scalable applications), the high cost of encrypting long messages in a asymmetric-key cryptography can be prohibitive.

A hybrid cryptography combines the convenience of an asymmetric-key cryptography with the efficiency of a symmetric-key cryptography. A hybrid cryptography can be constructed using any two separate cryptographic systems:

- A symmetric-key encapsulation scheme, which is a asymmetric-key cryptography.
- A data encapsulation scheme, which is a symmetric-key cryptography.

Thus a hybrid cryptography system is itself a public-key system, whose public and private keys are used to encapsulate symmetric keys. For data, the bulk of the work in encryption/decryption is done by the more efficient symmetric-key scheme, while the inefficient asymmetric-key scheme is used only to encrypt/decrypt a symmetric key value.

2.7 Conclusion

In this chapter, we gave a quick overview of the most used techniques for protecting data privacy.

First, we showed how access control techniques can help users in the system to control the access to their private data. Data access can be controlled with respect to the identity of the user, the role of the user and the access purpose for which a user requests data. Second, we presented Hippocratic databases which introduce the notion of purpose in relational databases. Then, we presented anonymity techniques which are important to protect user privacy. Furthermore, we described trust and reputation techniques. They can be used to predict in some way the behavior of users in the system and thus protect data privacy from malicious acts. Finally, we presented cryptographic schemes which can protect data privacy by making them unreadable by unauthorized users.

We assume that neither of these techniques does ensure alone data privacy in open environments. Combining these techniques provide more privacy guaranties. For instance, ensuring user anonymity makes it easier to malicious users to violate data privacy, in such circumstances trust techniques are needed. Sometimes it is very difficult to combine some techniques. For instance, access control who needs user identities can not be associated with anonymity techniques that hide users identity, in this case, smart cards or nicknames can be used.

We have shown the main characteristics of privacy techniques that can be used to achieve a particular level of privacy. The next chapter shows how these techniques are used in current P2P systems.

Chapter 3

Data privacy in P2P systems

The previous chapter showed privacy techniques and models that can be used in order to protect data privacy in information systems. This chapter shows how those techniques are used and adapted to data-centered applications based on P2P systems.

3.1 Introduction

In recent years, we have witnessed the massive ascend of the P2P paradigm in Internet-scale applications. P2P systems have been implemented more and more in diverse applications and services. They have been successfully used in services for sharing computation (e.g., Seti@home [ACK+02]), communication (e.g., Jabber [JAB]), Internet services (e.g., SopCast [SOP]) and data (e.g., Gnutella [GNU], Kazaa [KAZ]). In this work we focus on data-centered applications.

Several features distinguish P2P data-centered systems from traditional distributed database system:

- Peers are autonomous and free to join and leave the system anytime.
- P2P systems are scalable and can contain millions of peers.
- They make data available anytime and everywhere.
- They usually are open for every user interested in data sharing and storage.

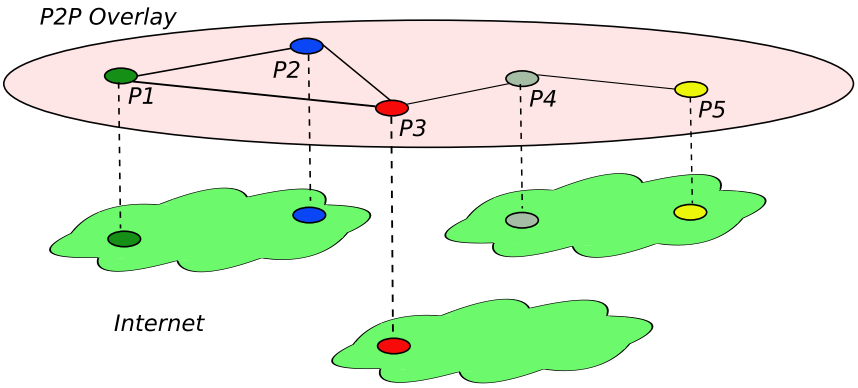


Figure 3.1: P2P overlay network on top of the Internet infrastructure

- There is no global schema for describing the data shared in the system.
- There is no centralized control over the data shared in the system.

Initial research on P2P systems has focused, among others, on improving performance on data publishing and searching. These works led to structured solutions based on Distributed Hash Tables (DHT), e.g., Chord [SMK+01], Pastry [RD01], Can [RFSH01] etc.

P2P systems rise new privacy concerns due to their openness. Currently, we witness new works that add privacy aspects to P2P systems. The goal is to ensure data and peers' privacy without affecting the P2P advantages.

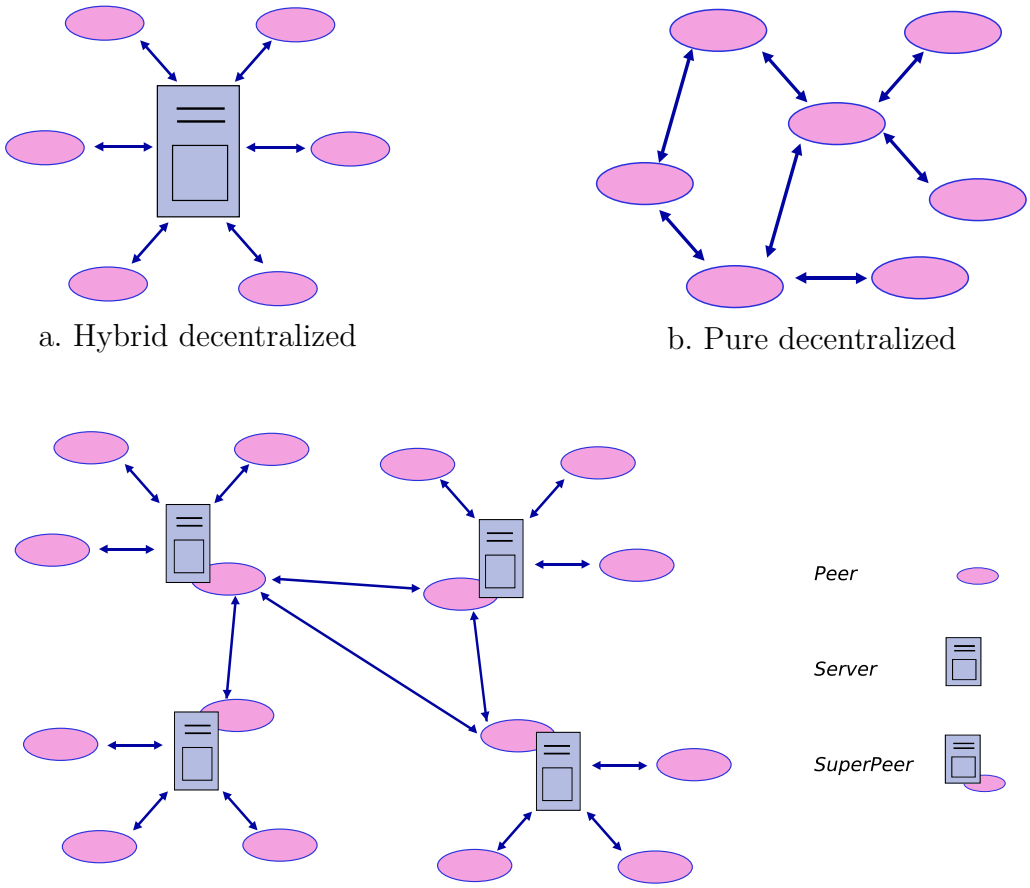
In this chapter, we survey techniques proposed for data privacy protection in P2P architectures. These architectures are compared based on several criteria: (a) which techniques are used to protect data privacy and (b) what sort of privacy protection do these architectures guarantee.

The rest of the chapter is organized as follows. Section 3.2 presents and discusses P2P systems. Section 3.3 presents data privacy management in current P2P architectures. Section 3.4 compares these architectures with respect to privacy concerns. Section 3.5 concludes.

3.2 Peer-to-peer paradigm

The P2P paradigm is a way to leverage vast amounts of computing power, storage and connectivity from personal computers distributed around the world [MKL+02]. It allows to scale up on a world wide scale without deploying expensive infrastructures like those needed for the client/server paradigm.

P2P systems operate on application-level networks referred to as overlay networks (cf., Figure 3.1). The degree of centralization and the topology of overlay networks have significant influence on properties such as performance, scalability and security. Many



a. Hybrid decentralized

b. Pure decentralized

c. Partially centralized.

Figure 3.2: Types of unstructured P2P overlays

works [BAH+06, LCP+05, ATS04] propose surveys of P2P systems and generally classify P2P networks into two main categories: unstructured (cf., Section 3.2.1) and structured (cf., Section 3.2.2).

3.2.1 Unstructured

Most popular P2P applications operate on unstructured networks. In these, the overlay network is created in an ad hoc fashion and data placement is completely unrelated to the organization of the overlay network. Each peer knows its neighbors, but does not know the resources they have. Query routing is typically done by *flooding* which consists on forwarding the query from a peer to all its neighbors and so on. Although P2P systems are supposed to be fully decentralized, in practice, three categories of unstructured networks can be encountered.

Hybrid decentralized architectures. In these networks (cf., Figure 3.2.a), a central server facilitates the interaction between peers by indexing all shared data. Queries are submitted to the server which knows the peer storing requested data. Then, the requester contacts directly the storer peer. Certainly, this approach provides an efficient data searching. However, the central server, which is a single point of failure, renders this architecture inherently unscalable and vulnerable to malicious attacks. A well-known example of hybrid decentralized architectures is Napster [NAP].

Pure decentralized architectures. In these networks (cf., Figure 3.2.b), there is no central coordination and all peers have equal roles. Each peer can issue queries, serve and forward queries to its neighbors (i.e., flooding). This approach provides peers dynamicity and there is no single point of failure. However guaranties on lookup efficiency can not be provided since peers knowledge about the system is limited to their neighbors. Representative examples of pure decentralized P2P systems are Gnutella [GNU], and FreeHaven [DFM00].

Partially centralized. In these networks (cf., Figure 3.2.c), super peers coordinate the interaction between other peers by indexing the data shared by peers connected to them (i.e., leaf peers). Queries are initially directed to super peers which serve as proxies of their leaf peers. Queries are then forwarded to leaf peers which store data. This approach provides a more efficient lookup in comparison with purely decentralized architectures while there still is no single point of failure. Representative examples of partially centralized systems are Kazaa [KAZ], and Edutella [NWQ+02].

Active research has improved performance on the unstructured P2P systems by reducing research costs [CFB04], reducing network traffic [APV06], insuring availability [CAMN03], etc.

3.2.2 Structured

Structured P2P networks have emerged to improve performances on data publishing and searching. They focus on introducing *structure* into the P2P network. They achieve this goal by controlling the overlay topology and the content placement. Aiming basically to act as a decentralized index, structured overlays provide a mapping between content (e.g., file identifier) and location (e.g., peer address), in the form of a distributed routing table.

Distributed hash tables. The most common type of structured P2P network is the one based on Distributed Hash Tables (DHT). DHT based systems support a distributed lookup protocol that efficiently locates the peer that stores a particular data item. Data location is based on associating a *key* with each data item, and storing the key/data item pair at the peer to which the key maps. To generalize, DHT provides two basic operations [DZD+03], each incurring $O(logN)$ messages.

- $put(k, data)$ stores a key k and its associated data object in the DHT.

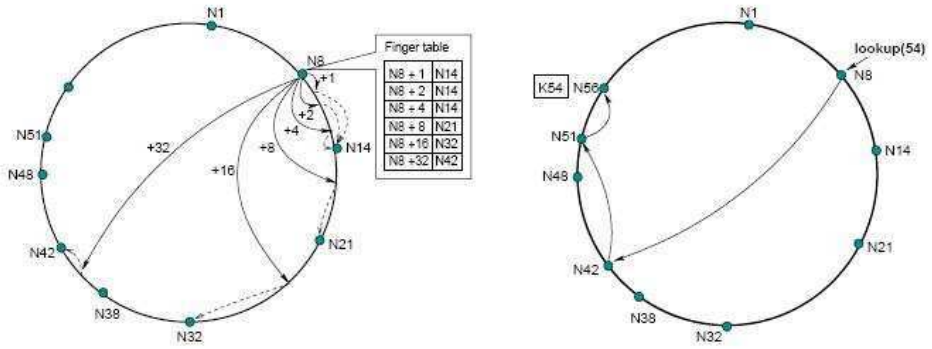


Figure 3.3: Chord DHT

- *get*(k) retrieves the data object associated with k in the DHT.

Because a peer is responsible of storing the values corresponding to a range of keys[1], autonomy is limited. Furthermore, DHT queries are limited to exact keyword search (e.g., file identifiers). Representative examples of DHTs are Chord [SMK+01] (cf., Figure 3.3), Pastry [RD01], Tapestry [ZHS+04] and Can [RFSH01]. Some examples of P2P systems that are based on DHT are OceanStore [KBC+00], Past [RH01], Freenet [CMH+02] and OneSwarm [IPKA09].

Active research tries to add new services above the DHT in order to extend their functionalities such as complex querying, availability, privacy protecting, etc. [RVLSA09] describes solutions for declarative querying support, query optimization and data privacy in DHT-based P2P systems and identifies important future research trends in data management in these systems. Next section shows how privacy techniques are integrated in P2P systems.

3.3 Data privacy in P2P systems

Providing privacy in P2P systems is a challenge due to their open and autonomous nature. In the literature, we have encountered many systems that propose solutions for different privacy issues in P2P systems. We categorize these systems into two main classes: those focusing on distributed data storage (cf., Section 3.3.1) and those focusing on massive data sharing (cf., Section 3.3.2).

[1]Responsibility for maintaining the mapping from keys to values is distributed among the peers, the way of distribution depends on the DHT used.

3.3.1 Privacy in distributed data storage systems

Distributed data storage systems are mainly used by users who want to benefit from large storage space and eventually share data with some peers. Usually, in this type of systems, users search for the following guaranties.

- Data are available anytime.
- Data are not read by unauthorized server peers.
- Data are not modified or deleted from the system.
- A peer does not claim the property of data owned by another peer.

Various techniques and protocols should be employed together to ensure such guaranties but also to provide privacy of content stored at potentially untrusted peers. For instance, replication can be used to guarantee data availability but replicas must be stored on trusted server peers. Thus, trust techniques and access control techniques can be used with replication in order to ensure data privacy. In addition, data digital checksums and encryption can be used to protect ownership rights and data from unauthorized reads.

Past [RH01], OceanStore [KBC+00], and Mnemosyne [HR02] are examples of systems that use those kind of techniques to preserve privacy in distributed data storage.

Past is a large-scale, Internet-based, global storage utility that provides scalability, high availability, persistence and security. It relies on Pastry [RD01] and uses smartcards, self certifying data and certified-based trust in order to protect data content from malicious servers.

OceanStore is a cooperative utility infrastructure that provides a consistent, highly-available, durable and secured storage utility. It relies on Tapestry [ZHS+04] and uses symmetric cryptography and access control techniques to protect data privacy from malicious peers.

Mnemosyne is a storage service that provides a high level of privacy by using a large amount of shared distributed storage to hide data. It relies on Tapestry and uses steganographic data, which presence among random data can not be detected, in order to protect data from malicious reads and deletion.

Section 3.4 compares these systems according to the privacy techniques used and privacy properties guaranteed.

3.3.2 Privacy in massive data sharing systems

Massive data sharing systems are mainly used by users who want (a) to publish data in the system in order to share them with a massive number of users and (b) to request and download data from the system. (b) is probably the main reason why this type of system is so well known and used especially in multimedia file sharing. In this type of system, users search for two types of guaranties:

1. Data privacy guaranties (in addition to those of Section 3.3.1)

- Data are available to every authorized requester peer.
- Data are not shared with unauthorized requester peers.

2. User privacy guaranties (usually referred as anonymity guaranties)

- Users are not monitored in the system by other peers.
- Users freedom of behavior is not limited by the system.
- Users are protected against identity theft.

Data privacy can be protected by techniques such as access control, trust management, data encryption and digital checksums. On the other hand, user privacy can be protected by using different anonymity techniques. However, ensuring user privacy may cause undesired effects on data privacy: users want to protect their data privacy and at the same time to remain anonymous and behave freely which increases the risk of violating data privacy of others. This loop is probably the main reason we did not find in the literature systems that guarantee both data and user privacy.

3.3.2.1 Protecting data privacy

Here, we introduce works focusing on data privacy in massive data sharing systems.

Office SharePoint Workspace [GRO], previously known as Office Groove, is a desktop application designed for document collaboration in teams (i.e., workspaces). It is based on a partially centralized P2P system. Each user has a privately editable copy of the workspace. Workspace copies are synchronized via the network in a P2P manner. Office SharePoint Workspace uses access control, trust and encryption techniques in order to protect data privacy.

Piazza [TIM+03] is a data management system that enables sharing XML documents in a distributed and scalable way. It is based on an unstructured P2P system. Although the goal and emphasis of Piazza is data sharing and not data privacy, the creators of Piazza proposed in [MS03] new techniques for publishing a single data instance in a protected form thus enforcing data privacy protection.

OneSwarm [IPKA09] is a P2P service that provides users with explicit control over their data privacy by letting them determine how data are shared. It relies on BitTorrent [BIT] and was designed to provide privacy-preserving data sharing. OneSwarm uses asymmetric cryptography, access control techniques, trust and communication anonymity in order to protect data privacy.

Other systems treated the censorship problem which can have effects on data privacy since censorship can be a reason for data suppression. With some systems such as Usenet news [USE], anyone who sees a message can post a cancel message to delete it. Many systems were proposed to resist to censorship. **Dagster** [SS01] is a censorship resistant publishing scheme that intertwine legitimate and illegitimate data from web pages, so that a censor can not remove *objectionable content* without simultaneously removing legally *protected content*. **Tangler** [WM01] is another censorship resistant publishing scheme

Application	P2P systems	Focus on	Relies on	Goals
Data storage	PAST	Protecting data privacy	Pastry	Scalable, highly-available, persistent and secure storage
	OceanStore		Tapestry	Consistent, highly-available, durable and secured storage
	Mnemosyne		Tapestry	Steganographic data storage
Massive data sharing	Office SharePoint Workspace		Partially centralized P2P	Document sharing, team collaboration
	Piazza		Unstructured P2P	Scalable XML sharing, data management system
	OneSwarm		BitTorrent	Privacy preserving data sharing
	Dagster		–	Censorship resistant data sharing
	Tangler		Own routing protocol	Censorship resistant data sharing
	Freenet	Protecting user privacy	Own routing protocol	Anonymous file sharing, freedom of speech
	SwarmScreen		BitTorrent	Privacy preserving data sharing

Table 3.1: Summary of the presented P2P systems

based on the idea of intertwining data. Newly published documents are dependent on previous published blocks. This dependency, called entanglement, provides a user with some incentive to replicate and store the blocks of other documents. Thus data blocks are resistant to censorship and suppression.

Censorship-resistant schemes protect data only from suppression. Data privacy is not fully protected since any user can access these data.

3.3.2.2 Protecting user privacy

Here, we present systems focusing on user privacy in massive data sharing. They mostly use anonymity techniques to guarantee anonymous publishing and sharing.

Freenet [CMH+02] is a free P2P system which ensures anonymous file sharing, browsing and publishing. It provides users with freedom of behavior by ensuring their anonymity. Freenet has its own key-based routing protocol, similar to DHT-based techniques. Freenet uses symmetric cryptography, user and communication anonymity in order to guarantee user privacy.

SwarmScreen [CDM+09] is a privacy preserving layer for P2P systems that disrupts community identification by obfuscating users' network behavior. SwarmScreen relies on BitTorrent and was designed to provide user privacy through plausible deniability. Since a user behavior can be deduced by its interests, SwarmScreen connects the user to other users outside of its community of interest which can disguise the user interests and thus its behavior.

Many systems such as **ANts P2P** [ANT] and **MUTE** [MUT] have been proposed as anonymous P2P file sharing softwares. They use anonymity in order to make the user untrackable, hide her identity and encrypt everything she is sending/receiving from others.

Censorship-resistant systems such as **Dagster** and **Tangler** usually use anonymity techniques to hide users identities. Thus, censorship on data belonging to a specific user

can not be done since the user identity is hidden.

Table 3.1 gives a summary of the presented P2P systems.

3.4 Comparaison

In this section, we compare the works presented in Section 3.3. Comparison focuses on the used privacy techniques (cf., Section 3.4.1) and the guaranteed privacy properties (cf., Section 3.4.2).

3.4.1 Used privacy techniques

Here, we compare the P2P systems presented in Section 3.3 with respect to access restriction, anonymity, trust and cryptography techniques. Table 3.2 resumes the techniques used in the compared P2P systems. In this table, a cell is kept blank when a privacy technique is not used by the P2P system. *N/A* is used when information about a privacy technique is not available in the literature.

3.4.1.1 Access restriction

The use of access restriction is primordial to guarantee that data will not be read or shared with unauthorized peers.

In **OceanStore**, access control is based on two types of restrictions: *reader* and *writer* restrictions. In the *reader* restriction, to prevent unauthorized reads, the data decryption keys are distributed by the data owner to users with read permissions. To revoke read permissions, the data owner requests to users to delete replicas or re-encrypt them with new encryption keys. A recently-revoked reader is able to read old data from cached copies or from misbehaving servers that fail to delete or re-encrypt. This problem is not specific to OceanStore, even in conventional systems, there is no way to force a reader to forget what has been read. To prevent unauthorized writes, they must be signed so that well-behaved servers and clients can verify them against an access control list (ACL). The data owner can define the ACL for data by providing a signed certificate. ACLs are publicly readable so that servers can check whether a write is allowed. Thus, writes are restricted at servers by ignoring unauthorized updates.

In **Piazza**, the access to a published XML document is restricted to parts of the document in accordance with the data owner preferences. Data owners in Piazza can specify access control policies declaratively and generate data instances that enforce them. By granting decryption keys to users, the data owner enforces an access control policy. Once published, the data owner relinquishes all control over who downloads and processes the data. Requesters can access the data conditionally, depending on the keys they possess.

In **OneSwarm**, persistent identities allow users to define per-file permissions. These permissions (i.e., capabilities) restrict access to protected data. For example, OneSwarm can be used to restrict the distribution of a photo file to friends and family only.

In **Past**, access control is based on the use of smartcards which generate and verify various certificates. Users may access data or not within the access rights related to their certificate.

In **Office SharePoint Workspace**, access control is based on the use of membership lists and workspace rules. Users are identified by accounts and own passwords that allow them to log in workspaces. If they can log in a workspace W, they are listed in the membership list of W. A user can access or remove data from W as long as she is a member of W and, in addition, respects the workspace usage rules.

3.4.1.2 Anonymity techniques

Anonymity can enable censorship resistance, freedom of behavior without fear of persecution, and privacy protection (cf., Section 2.4). Anonymity is mostly used to hide user identity. If user identity is hidden, access control techniques could not be deployed. On the other hand, if anonymous communication channels are used, a channel listener is not able to understand the messages sent on the channel or who has send it.

In **Freenet**, privacy is maintained by using anonymous communication. Rather than moving directly from sender to recipient, messages travel through peer to peer chains, in which each link is individually encrypted, until the message finally reaches its recipient. Each peer knows only about its immediate neighbors thus the end points could be anywhere among the network's peers, which are continually exchanging indecipherable messages. Not even the peer immediately after the sender can tell whether its predecessor was the message's originator or was merely forwarding a message from another peer. Similarly, the peer immediately before the receiver can not tell whether its successor is the true recipient or will continue to forward it.

In **Dagster**, privacy is also maintained by using anonymous communication. An anonymous channel between owners or requesters and servers is created by using the Anonymizer[2]. Instead of requesting web data directly, a user sends the request to the Anonymizer which forwards the request appropriately. The content is then delivered to the Anonymizer which returns it to the requesting user. The Anonymizer can only be used to retrieve data content and the user is required to trust the Anonymizer's operators not to reveal her identity.

In **Tangler**, privacy is maintained by using anonymous communication and identities. Tangler ensures that a user can retrieve data without revealing their identity. Users publish documents by anonymously submitting blocks to servers. Servers can communicate with each other both directly and anonymously (by using other servers as a mixed network [Cha81]).

In **SwarmScreen**, privacy is maintained by using random connections. They propose a privacy-preserving layer for P2P systems that disrupts community identification by obfuscating users' network behavior. Users can achieve plausible deniability by simply

[2]The Anonymizer is an online service that attempts to make activity on the Internet untraceable. It accesses the Internet on the user's behalf, protecting personal information by hiding the source identifying information. http://www.anonymizer.com/.

adding a particular percent (between 25 and 50 %) of additional random connections that are statistically indistinguishable from natural ones.

In **OneSwarm**, anonymity is only used to protect a user identity from a third-party monitoring. Users in OneSwarm perform their queries by using anonymous routes. However, a server has the complete knowledge of the query initiator identity, thus access control is possible.

Another way to use anonymity is to give users fake identities. Fake identities can be ensured by smartcards techniques where the real identity of the user is only known by the authority which distributes the smartcards. In this case, the authority must be considered as trustworthy. In **Past**, smartcards are used to allow users to obtain necessary credentials to join the system in an anonymous fashion.

Systems like **Office SharePoint Workspace** organize users into anonymous groups called workspaces. Users are known within their workspace and they are anonymous for users in other workspaces. This choice can be explained by the fact that users do not need to be anonymous to their friends or to co-workers who are authenticated in order to access their workspace.

3.4.1.3 Trust techniques

Trust techniques are used in P2P systems in order to prevent data privacy violation. The right to access data can be given to peers who are trustworthy and forbidden to peers who are untrustworthy (cf., Section 2.5).

Mainly, P2P systems that preserve privacy in distributed data storage do not trust data servers. The potential malicious behavior of servers can be faced with cryptography techniques. The systems analyzed here use trust techniques to verify trustworthiness of peers who want to access data securely stored on servers.

In **Past**, server peers trust owner peers thanks to a smartcard held by each peer which wants to publish data in the system. Smartcards are given by authorities called brokers who are fully trusted by owner and server peers. A smartcard ensures the integrity of identifiers and trustworthiness assignment of the user which held it. Without a trustworthy third party, it is difficult to prevent attackers from misbehaving in the system.

On the other hand, in systems that provide massive data sharing, data owners are usually trustworthy and cryptographic techniques are used to face the malicious behavior of servers. However these systems are more interested in verifying trustworthiness of requesters who want to access private data.

In **Office SharePoint Workspace**, peers can determine how much to trust other peers through their authentication status. A peer A can optionally organize her contacts by how they were authenticated or check their authentication status by the color of their name. The names of directly authenticated contacts, which are trustworthy, are displayed in green. Other contacts in A's workspace, which are also trustworthy, are displayed in teal. Contacts in other workspaces trusted by the A's domain administrator are displayed in blue. Contacts that are not authenticated are displayed in black, and duplicated names that conflict are displayed in red. The color of the name can be used in the verification

of trustworthiness of the requester. Requesters who have their name in black or red are considered untrustworthy and thus they may not gain access to data.

In **OneSwarm**, data are located and transferred through a mesh of untrusted and trusted peers populated from user social networks. Peers explicitly define a trust level for a persistent set of peers. This requires some notion of identity to allow peers to relate real-world trust relationships to overlay connections. Public keys can be used as identities in order to verify trustworthiness of the peers. These public keys can be exchanged in three ways: first, requesters discover and exchange keys with owners over the local area network. Second, peers can rely on existing social networks, e.g., Google Talk or Facebook, to distribute public keys. Third, peers can email invitations to friends. Invitations include a one-time use capability that authenticates the recipient during an initial connection, during which public key exchange occurs. OneSwarm also supports key management within a group. It allows peers to subscribe to one or more community servers. A community server maintains a list of registered peers and can delegate trust regarding a subset of their peers.

3.4.1.4 Cryptography techniques

Cryptography is largely used by P2P systems in order to protect private data from unauthorized access. Encryption techniques are used to prevent malicious servers from reading private data, while digital checksums are used to detect if malicious peers are modifying or corrupting private data.

Data encryption. Usually symmetric-key encryption (cf., Section 2.6.1) is used to protect data content. Symmetric key generation is less expensive than asymmetric keys generation. Since a large number of keys is needed to encrypt data, P2P systems have found more interest in symmetric-key encryption.

In **OceanStore**, data are encrypted using symmetric keys. Encryption keys are distributed to users who are allowed to access data.

In **Piazza**, published data are encrypted in order to restrict peers from accessing data in accordance with the owners' preferences.

In **Freenet**, all data are encrypted before publication. This is done majorly for political or legal reasons where servers might wish to remain ignorant of the content of the data stored. Data encryption keys are not included in network messages. Owners distribute them directly to end users[3] at the same time as the corresponding data identifiers. Thus, servers cannot read their own files, but users can decrypt them after retrieval.

In **Office SharePoint Workspace**, data are encrypted on the communication channels. Data that may temporarily be stored on servers are also encrypted using keys kept by owners, preventing potentially malicious servers from reading data. However, a

[3]Freenet does not use access control techniques thus keys distribution is not restricted to authorized requesters.

user has the choice to delegate her identity management to servers hosted by Microsoft[4] or a third party. If so, this one will have access to the encryption keys.

Other systems like Mnemosyne, Dagster and Tangler use block encryption (cf., Section 2.6.1).

In **Mnemosyne**, data are divided into blocks, then to store data each block is encrypted using SHA256 and AES and is written to a pseudo-randomly chosen location. With a good enough cipher code and key, the encrypted blocks will be indistinguishable from the random substrate, and so an attacker cannot even identify the data. On the other hand, users who have the data name and key can reconstruct the pseudo-random sequence, retrieve the encrypted blocks, and decrypt them.

In **Tangler**, data are broken into a number of small blocks (shares). Each of these blocks is treated independently and stored on a subset of the participating servers. Blocks are then entangled with other random blocks which obscures the real content of the block making it unreadable. In order to reconstruct a data block, users have to retrieve a minimum number of blocks of the appropriate shares. By simply stripping away the random value, users can find the original data block.

In **Dagster**, data are separated into blocks, then the user generates a symmetric key for each block. Each block is encrypted with the corresponding key and sent to servers using anonymous channels. In order to reconstruct data, a number of blocks is needed along with the decryption keys.

Other systems, such as **OneStorm**, combine public key encryption with symmetric encryption. While symmetric keys are used to encrypt data, public keys are used to share symmetric keys in a secure manner.

Data integrity protection techniques. Having private data encrypted prevents unauthorized peers from reading their content. This contributes to protect data content privacy, however it does not protect data from being corrupted or deleted.

To protect data from suppression and corruption, cryptographic hash functions (digital checksums) can be used (cf., Section 2.6.1).

In **OceanStore**, the data are named using a secure hash over the data content, giving them globally unique checksums. This provides data integrity, by ensuring that requested data have not been corrupted, since the checksum of corrupted data will be different than the globally unique checksum.

In **Freenet**, when a user publishes data which she later intends to update, she first generates a public-private key pair and signs the data with the private key. Data are published under a pseudo-unique binary key (i.e., hash key), but instead of using the hash of the data contents, the data identifier itself is used (a signature-verifying key). Signature-verifying keys can be used to verify that the data contents have not been tampered with.

[4]Microsoft kept their right to collect some information about use of the Office SharePoint Workspace software and other activities "outside" of workspaces, as explained in their privacy statement at http://office.microsoft.com/en-us/help/privacy-supplement-for-microsoft-office-groove-2007-HA010085213.aspx

In **Past**, a smartcard user generates reclaim certificates. This certificate contains the data identifier, it is signed by the smartcard and included in the user request. When processing a request, the smartcard of a server peer first verifies that the signature in the reclaim certificate matches the one in the data certificate stored with the data. This prevents unauthorized users from reclaiming the ownership of data.

3.4.2 Guaranteed privacy properties

The use of privacy techniques as presented in the previous section allows P2P systems to guarantee the following privacy properties:

- Data protection against unauthorized reads. Unauthorized server peers should not have the ability to read data they store.
- Data protection against corruption and deletion. Unauthorized server peers should not have the ability to corrupt or delete data they store.
- Limited disclosure. Data should not be disclosed to unauthorized peers.
- Anonymity. Data users should not be identified.
- Denial of Linkability. Peers should have the ability to deny the links they had with other peers.
- Content deniability. Peers should have the ability to deny their knowledge on data content.

Tables 3.3 and 3.4 resume the privacy properties guaranteed in the compared P2P systems. In these tables, a cell is kept blank when a privacy property is not guaranteed by the P2P system. *N/A* is used when information about a privacy property is not available in the literature.

3.4.2.1 Data protection against unauthorized reads

In order to protect data from unauthorized reads data encryption is used. Data encryption prevents server peers, and eventually malicious eavesdroppers and routing peers, from reading private data.

OceanStore, Piazza, Mnemosyne, Freenet, Dagster, Tangler, Office SharePoint Workspace and **OneSwarm** guarantee data protection from unauthorized reads by using data encryption. In **Mnemosyne, Freenet, Dagster, Tangler** and **SwarmScreen** data are public. In **Past**, servers are not controlled and data are not encrypted, thus data are not protected against unauthorized server reads.

3.4.2.2 Data protection against corruption and deletion

There is no technique that prevents data from being corrupted or deleted. However, data integrity techniques and digital checksums can be used to help users to verify if the data have suffered an unauthorized modification.

OceanStore and **Freenet** use digital checksums to verify that the data content has not been tampered with. **Past** uses smarcards to sign data in order to authenticate data and verify if they have been modified or corrupted. Although private data in these systems are not protected against corruption and deletion, malicious modifications can be detected and countermeasures can be taken against malicious peers in order to prevent any future corruption. In other systems, data checksums are not used thus data modifications can not be detected.

3.4.2.3 Limited disclosure

In order to limit data disclosure, access control is used. It prevents unauthorized requesters from accessing the data. It is also used in distributed storage systems to prevent owners from accessing other peers' data.

In addition to access control techniques, encryption is used to prevent unauthorized disclosure due to collusion between servers and requesters. Even if a server discloses encrypted data to an unauthorized requester, she could not access data content without having the decryption keys.

In addition, trust techniques can be used. Owners will be more assured about the use of their data when they can verify the trustworthiness of the requester.

Past and **OceanStore** use access control to limit disclosure only for authorized data owners[5]. **Freenet, Dagster** and **Tangler** do not use access control techniques since the peers are anonymous. Thus disclosure are unlimited and not controlled. **Piazza** uses access control techniques to limit disclosure for authorized users. **Office SharePoint Workspace** and **OneSwarm** both use access control and trust techniques, thus, data disclosure is limited not only to authorized peers but also to trustworthy ones.

3.4.2.4 Users anonymity

[DGMY03] defines four types of anonymity guarantees: 1) author (i.e., owner) anonymity (which users created which documents?), 2) server anonymity (which peers store a given document?), 3) reader (i.e., requester) anonymity (which users access which documents?) and 4) document anonymity (which documents are stored at a given peer?).

PAST guarantees author and server anonymity due to pseudonymity techniques. Each user holds an initially unlinkable pseudonym in the form of a public key. The pseudonym is not easily linked to the user's real identity. If desired, a user may have several pseudonyms to hide that certain operations were initiated by the same user. PAST users do not need to reveal their identity, nor the data they are retrieving, inserting or storing.

[5]These systems are not meant for massive data sharing, thus information about data disclosure for requesters is not available.

Freenet guarantees author, server and reader anonymity due to anonymous communication. Author anonymity is protected by occasional resetting of the data source field in response messages. The peer appearing as the data source does not imply that it actually supplied that data. For reader and server anonymity, while a peer can get some indication on how early the request message is on the forwarding chain using the limit on the number of hops (hop-to-live), the true reader and server are kept private due to anonymous communication.

Dagster and **Tangler** guarantee author, server and reader anonymity due to anonymous communication and anonymous identities. Because all connections between the server and the owner/requester are over an anonymous channel, there is no correlation between their identities and the documents that they are publishing or requesting.

SwarmScreen and **OneSwarm** guarantee author and reader anonymity due to anonymous communication but only from third-party monitoring. In OneSwarm, users' identities are known by servers in order to perform access control thus their anonymity is not guaranteed. Server anonymity is also not guaranteed because they must be easily located in order to publish or request data.

Office SharePoint Workspace guarantees author, server and reader anonymity due to anonymous workspaces. We recall that inside workspaces anonymity is not preserved.

Document anonymity is discussed in Section 3.4.2.6.

3.4.2.5 Denial of linkability

Linkability refers to identifying the correlation between users. Knowledge on linkability can be denied by using anonymous communication. Denial of linkability can protect peers form third-party monitoring.

Freenet, Dagster, Tangler, OneSwarm and **SwarmScreen** guarantee denial of linkability due to anonymous communication. **Past** uses pseudonymity techniques, thus knowledge on linkability may be denied by peers. **Office SharePoint Workspace** guarantees denial of linkability due to anonymous workspaces.

3.4.2.6 Content deniability

Content deniability refers to whether peers can deny the knowledge on the content stored or transmitted (document anonymity). Knowledge on content can be denied if the content is not readable by the peer that hold it.

In all systems that use data encryption, knowledge on data content can be denied since data are unidentifiable. This is the case of **OceanStore, Mnemosyne, Piazza, Freenet, Dagster, Tangler, Office SharePoint Workspace** and **OneSwarm**. In **Past** and **SwarmScreen**, servers can access data thus they can not deny the knowledge on the data content.

P2P Systems	Privacy Techniques				
	Access control techniques	Anonymity	Trust techniques	Data encryption	Data integrity protection
PAST	Use of smartcards	Pseudonymity using smarcards	Certificate-based Trust		Use of smartcards for data certification
OceanStore	Use of two types of restriction (reader and writer)			Symmetric encryption	Use of content hashing as a checksum
Mnemosyne				Symmetric block encryption	
Office SharePoint Workspace	Use of workspace and membership lists	Anonymous workspaces and groups	Use of trust colors	Symmetric encryption	
Piazza	Use of access policies defined by the owner			Symmetric encryption	
OneSwarm	Use of permissions defined by the owner	Anonymous communication	Use of community trust levels	Hybrid encryption	
Dagster		Anonymous communication		Symmetric block encryption	
Tangler		Anonymous communication and identities		Symmetric block encryption	
Freenet		Anonymous communication		Symmetric encryption	Use of signature-verifying keys
SwarmScreen		Random communication			

Table 3.2: Comparison of P2P systems based on used privacy techniques

3.5 Conclusion

In this chapter, we discussed data privacy techniques used by representative P2P systems.

First, we introduced the types of P2P networks with respect to network topologies. Then, we briefly described some of the well known P2P systems. We organize systems in two types : P2P distributed storage systems and P2P massive data sharing systems.

We compared these systems based on the privacy techniques used to protect data privacy (cf., Table 3.2) and the privacy properties guaranteed (cf., Tables 3.3 and 3.4). We recall that, in the comparison tables, a cell is kept blank when a privacy technique (resp. property) is not used (resp. guaranteed) by the P2P system. *N/A* is used when information about a privacy technique or property is not available in the literature.

To resume, the following techniques were investigated in the comparison:

- Access control techniques which prevent unauthorized and malicious access.
- Anonymity techniques which can hide information about users and data.

P2P Systems	Guaranteed properties			
	Protection against unauthorized reads	Protection against corruption and deletion	Limited disclosure	
			Owners	Requesters
PAST		Yes due to data digital checksums	Yes due to access control (smartcards)	N/A
OceanStore	Yes due to encryption	Yes due to data digital checksums	Yes due to access control	N/A
Mnemosyne	Yes due to encryption			
Office SharePoint Workspace	Yes due to encryption		Yes due to access control	Yes due to access control
Piazza	Yes due to encryption		Yes due to access control	Yes due to access control
OneSwarm	Yes due to encryption		Yes due to access control	Yes due to access control
Dagster	Yes due to encryption			
Tangler	Yes due to encryption			
Freenet	Yes due to encryption	Yes due to data digital checksums		
SwarmScreen				

Table 3.3: Comparison of P2P systems based on guaranteed privacy properties

- Trust techniques which predict peers behavior and help peers distinguishing trustworthy peers from untrustworthy ones.
- Data encryption and integrity protection techniques which protect data privacy from malicious reads, writes, corruption and deletion.

In addition, we showed how the compared P2P systems may guarantee:

- Protection against unauthorized reads. by using data encryption.
- Protection against corruption and deletion.
- Limited disclosure.
- Anonymity and denial of linkability.
- Content deniability.

The comparison shows that existing P2P systems do not use completely all the privacy techniques, and neither of them guarantees all the privacy properties at once. The main reasons for this can be resumed to either cost optimization issues or differences in definitions for privacy protection.

Some systems lack basic mechanisms for supporting trust techniques because:

P2P Systems	Guaranteed properties				
	Anonymity			Denial of linkability	Content deniability
	Authors	Servers	Readers		
PAST	Yes due to Pseudonymity	Yes due to Pseudonymity	Yes due to Pseudonymity	Yes due to Pseudonymity	
OceanStore					Yes due to encryption
Mnemosyne					Yes due to encryption
Office SharePoint Workspace	Yes due to workspaces anonymity	Yes due to workspaces anonymity	Yes due to workspaces anonymity	Yes due to workspaces anonymity	Yes due to encryption
Piazza					Yes due to encryption
OneSwarm	Only for third-party monitoring		Only for third-party monitoring	Yes due to communication anonymity	Yes due to encryption
Dagster	Yes due to anonymous communication	Yes due to anonymous communication	Yes due to anonymous communication	Yes due to anonymous communication	Yes due to encryption
Tangler	Yes due to anonymous identities	Yes due to anonymous identities	Yes due to anonymous identities	Yes due to anonymous communication	Yes due to encryption
Freenet	Yes due to anonymous communication	Yes due to anonymous identities	Yes due to anonymous identities	Yes due to anonymous communication	Yes due to encryption
SwarmScreen	Yes due to anonymous communication	Yes due to anonymous communication	Yes due to anonymous communication	Yes due to anonymous communication	

Table 3.4: Comparison of P2P systems based on guaranteed privacy properties - continued

- Trust techniques can be very expensive in term of sent messages which can impact scalability. Although these systems privilege scalability and performance, they are vulnerable to malicious data access, and data privacy can be easily violated by untrustworthy peers. Finding optimal trust techniques that introduce small acceptable overhead can be a solution for these systems.

- Some systems like Freenet, Dagster and Tangler prefer to focus on problems such as censorship-resistant data sharing or user anonymity rather than protecting data privacy from misbehaving peers.

Some systems use anonymity techniques to protect user privacy, but data privacy is not protected because, with anonymous users, identity control techniques could not be employed. Thus, it is not surprising to see that systems like Freenet, Dagster, Tangler and SwarmScreen, do not use access control techniques because they employ anonymity techniques. These systems are vulnerable to unauthorized access. Data privacy can be easily violated by malicious peers which have all the freedom to access whatever data they want.

We have not found in the literature any P2P system that uses Hippocratic databases, more precisely purpose specification, that is why this privacy technique was not considered in the comparison. Although purpose specification is essential to privacy protection as

accentuated by the OECD guidelines, Hippocratic principles and P3P specifications, it has not been used yet in P2P systems. Thus a new privacy solution for P2P systems that uses purpose-based access control is needed. The next chapter presents PriMod, a privacy model that focus on access control, purpose specification, trust management and cryptography to protect data privacy.

Chapter 4

PriMod, a data privacy model for P2P systems

This chapter presents PriMod, a data Privacy Model for P2P systems that focuses on definitions and concepts to guarantee the data protection against unauthorized reads, data protection against corruption and deletion, purpose specification, limited disclosure and content deniability.

4.1 Introduction

Access control, trust, cryptography and other techniques can be combined together to protect data privacy in P2P data management systems (cf., Chapters 2 and 3). However, even if the notion of purpose has a relevant importance (cf., Chapters 1 and 2), it has not been applied yet in P2P systems.

The necessity of purpose specification motivates the proposition of a new purpose-based privacy model for P2P systems.

PriMod is a data privacy model for P2P systems that combines purpose-based access control, trust and cryptographic concepts. PriMod is proposed to answer the need of data owners to share their sensitive data in a P2P system while preserving data privacy.

In PriMod, peers may be participants who share data, request information, or simply contribute to the storage system. We consider that there are peers who own data and that do not necessarily act as servers of those data. Thus we distinguish three kinds of peers:

- **Requester.** A peer that requests data.
- **Server.** A peer that stores and provides data.
- **Owner.** A peer that owns and shares data.

PriMod makes no assumptions about the P2P system organization, it can be structured, semi-structured or unstructured. The unique important hypothesis taken is that each peer has a unique identifier in the system for all its connections. [CDG+02, RBTM07] have treated peer identification[1].

We consider that peers misbehave if they violate data privacy preferences defined by data owners. In this model we concentrate on preventing:

- **Unauthorized disclosure.** Server peers can misbehave by disclosing data to requester peers who, based on the owner data privacy preferences, should not have those data.
- **Data misuse.** Requester peers misbehave if they do not respect agreements made during data access.
- **Attacks to data integrity.** Server peers may violate data integrity by modifying data content of data they provide without permission.

PriMod allows owners to choose between publishing only their data references (e.g., filenames, primary keys, etc.) or publishing encrypted data content. Requesters can search for sensitive data but must specify the access purpose and operation in their requests, thus they are committed to their intended use of data.

PriMod also allows peers to know which sensitive data they can access for a particular purpose. To motivate this, we consider scientists who wish to research on a particular disease. According to the article L1122-1 of the French public health code[2], prior to conducting any biomedical research on a person, the scientist must have the consent of this person, or the doctor who represents him, on the use of the sensitive data, in particular the purpose for which data will be used. Thus, the scientist needs to contact the health care department to have the consent of all the doctors on the use of sensitive data needed to conduct research. PriMod offers a simple way to have this consent: the health care organization will send requests (private letters) to the doctors concerned by this research and ask them to publish all data references that the scientist can access for particular purpose and operation. By requesting the list of these references, the scientist can be informed of the data she can access and use for her research.

To resume, PriMod assets are the following:

- It benefits from P2P assets in data publishing and sharing while offering data privacy protection.

[1] We are fully aware of the impact of identification on user privacy. However, peer identities do not necessarily reveal users real identities thus user privacy can be protected.

[2] www.legifrance.gouv.fr

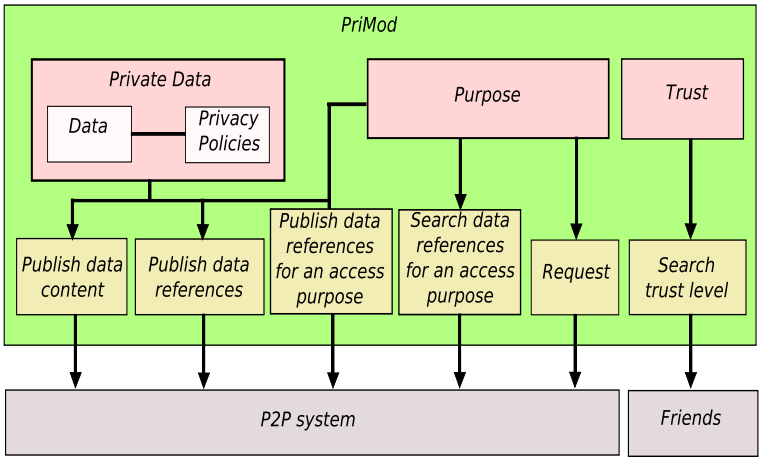


Figure 4.1: PriMod

- It can be easily integrated to any P2P system.
- It proposes/uses concepts and models for privacy policies, trust, and data, and offers operations for publishing data content, publishing references, requesting, and purpose-based searching (cf., Figure 4.1).
- Data owners can define their privacy preferences in privacy policies.
- Sensitive data are associated with privacy policies.This association creates private data ready to be published in the system.
- Requests are always made for particular purposes and operations.
- Trust techniques are used to verify trustworthiness of requester peers.

This chapter is organized as follows. Section 4.2 presents the purpose model. Section 4.3 presents the privacy policies model. Section 4.4 shows the used trust model. Section 4.5 presents the used data model. Section 4.6 introduces the functions of PriMod. Section 4.7 concludes.

4.2 Purpose model

PriMod uses the purpose model presented in [BL08].

Purposes. They describe the reasons for data collection and data access. They are organized in a purpose tree PT.

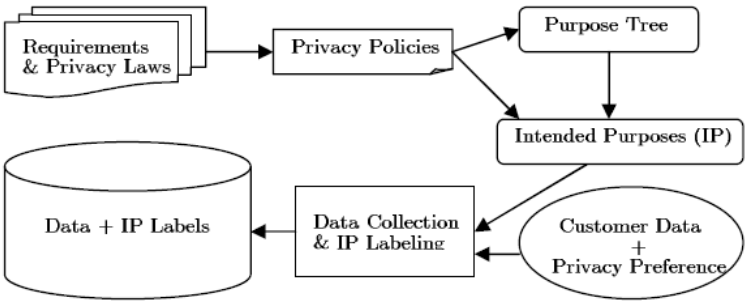


Figure 4.2: Intended purpose management process

Specialization/generalization. The relationships between purposes are specialization/generalization relationships. If p1 and p2 are two nodes in *PT* tree, and there is an edge <p1,p2>, then p2 is a specialization of p1 (or p1 is a generalization of p2).

Intended purposes. They specify the usages for which data are collected. An intended purpose *IP*, is a tuple *<AIP, PIP>*, where *AIP* is a set of allowed intended purposes and *PIP* is a set of prohibited intended purposes.

Access purpose compliance. An access to a specific data item is allowed if *IP* allowed by privacy policies for the data include or imply the access purpose *AP*. That is, an access is granted if the access purpose is entailed by the allowed intended purposes *AIP* and not entailed by the prohibited intended purposes *PIP*. The access is denied if any of these two conditions fails. Figure 4.2 shows the intended purpose management process.

4.3 Privacy policy model

In PriMod, each data owner should define its privacy preferences. Those data privacy preferences are registered in privacy policies (PP). PPs are defined independently of data. Once defined, they are attached to appropriate data. PPs are dynamic because they can vary with time. For instance, at the end of the cure, a doctor will only allow reading access to other doctors for analyzing the patient medical record. Updating of diagnosis will not be allowed anymore. We consider that each owner is responsible for defining and maintaining her privacy policies in an independent way. Inspired from P3P [CLM+02], Figure 4.3 shows a PP model. This model does not claim to be exhaustive. It shows information about PPs which can include:

Authorized Users. It is a list of users who are authorized to access data, a kind of ACL. A user can be an individual or a group.

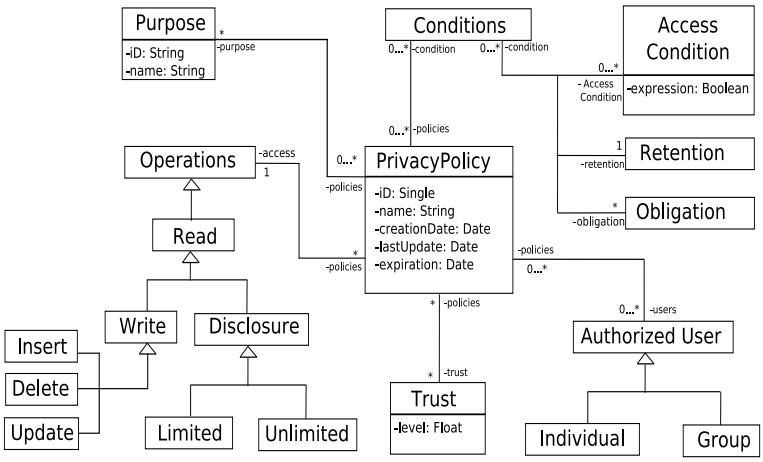


Figure 4.3: Privacy policy (PP) model

Purpose. An access purpose states the data access objective. With this concept, an owner is able to specify the purposes for which its data can be accessed by users.

Operations. An operation determines what a peer can do with data. We use three basic operations, read, write and disclose, but others can be defined.

- **Read.** A peer can read the data content.
- **Write.** A peer can modify the data content with the following operations: insert, update, delete.
- **Disclose.** A peer is able to disclose shared data for other peers. Disclosure can be limited or unlimited. If a peer disclosure right is limited, it can not give disclosure rights on the data to other peers.

Conditions. Conditions state the access conditions a user should respect, the obligations a user should accomplish after accessing data and the limited time for data retention.

- **Access condition.** Conditions state under which semantic condition data can be accessed. This may concern data values, for example age>10.
- **Obligation.** Obligations state the obligation a user must accomplish after the data access. For example, researcher Ri should return research results after using the age value of patient x.
- **Retention time.** The retention time states the time limit of retention of the data. For example, the age value of patient x should be destroyed after 10 days of use.

Minimal trust level. It is the minimal trust level a requester peer should have in order to gain access to data.

4.4 Trust model

Trust is a fuzzy concept where trusted requesters are supposed to respect privacy policies defined by the data owners. Trust between peers is important in order to prevent data access violation. The trust model we use shows information about the trustworthiness of peers. It includes:

Trust level. The trust level reflects a peer trustworthiness with respect to other peers. A peer can have different trust levels at different peers. We consider that trust levels vary in a range of [0,1].

- An **honest peer** is a peer which has a high trust level e.g., in a range of [0.5, 1].
- A **malicious peer** is a peer which has a low trust level e.g., in a range of [0, 0.5).
- An **unknown peer** is a peer which has an unknown trust level from the owner point of view.

Friends. A *friend* of a peer P is a peer with a high trust level from P's point of view. The number of friends held by a peer can vary from one peer to another. A peer can locally have the trust levels of friends and some peers which have interacted with it. If a peer does not have a particular trust level, it can ask for this to its friends. The trust level received will be weighted with the friends trust levels.

Addition of new peers. When a new peer joins the system, it is considered as an unknown peer. Peers are free to assign its trust levels based on direct interactions.

4.5 Data model

In order to respect PPs, we define a specific data model where they are associated with data. In the following, we use relational tables. However our model can consider any type of data (files, XML documents, rich text files, etc.).

Following the motivating example showed in the introduction (cf., Section A.1), we consider a collaborative medical research focusing on the evolution of cardiovascular diseases. The participants of this research are scientists, doctors, patients, etc. The privacy of sensitive data such as medical records owned by doctors or research results owned by scientists, should be protected according to the privacy preferences of data owners.

In the next, we present data tables (cf., Section 4.5.1), privacy policy tables (cf., Section 4.5.2), purpose tables (cf., Section 4.5.3), trust tables (cf., Section 4.5.4), private data

Data table DTj							
Id (PK)	**SS**	**Name**	**Country**	**Birthdate**	**Gender**	**Smoker**	**Diagnosis**
Pat1	001044001001	Alex	France	2000	Male	No	NO cardiovascular disease
Pat2	900344001001	Bea	France	1990	Female	No	NO cardiovascular disease
Pat3	730844001001	Chris	France	1973	Male	No	YES cardiovascular disease
Pat4	441144001001	Dave	Belgium	1944	Male	Yes	YES cardiovascular disease
Pat5	680544001001	Elena	Russia	1968	Female	Yes	YES cardiovascular disease

Table 4.1: Data table of doctor Dj

Privacy policies table PPTj					
Id (PK)	**Operation**	**User**	**Purpose**	**Condition**	**Minimal trust level**
PP1	Read	Pharmacists, Doctors	Consulting record	Birthdate < 2000	0.5
PP2	Read	Researchers	Researching on cardiovascular disease	—	0.6
PP3	Write	Dk	Second diagnosis	—	0.9

Table 4.2: Privacy policies table of doctor Dj

tables (cf., Section 4.5.5) and data reference tables required for purpose-based searching (cf., Section 4.5.6).

4.5.1 Data table

We consider that each owner peer stores locally the data it wants to share. Those data can be stored in relational tables which we call *data tables*. The unique restriction about data tables is that primary keys should be generic and impersonal to respect privacy and to not disclose any information. If considered data are files, their identifiers or names should be impersonal. Following our motivating example, Table 4.1 shows the medical records of doctor Dj.

Purpose table PTj		
PurposeId	**Purpose Name**	**Purpose Description**
Purpose1	Consulting record	Patient records are only read for consulting
Purpose2	Researching on cardiovascular disease	Data are only used for researching on cardiovascular disease
Purpose3	Second diagnosis	Data are only used for second diagnosis

Table 4.3: Purpose table of doctor Dj

Trust table TTj		
RequesterId	**Trust Level**	**Friend**
P54	0.85	Yes
P18	0.72	No
P72	0.10	No

Table 4.4: Trust table of doctor Dj

4.5.2 Privacy policy table

Data contained in PPs are stored in a table named *privacy policies table*. To simplify, we do not include all elements of Figure 4.3. In this table, one tuple corresponds to one PP. The same PP can be used with different data. Each policy contains operations (read, write, disclose), allowed users, access purposes, conditions (if they exist), and the required minimal trust level of allowed users. Table 4.2 shows the privacy policies table of doctor Dj.

4.5.3 Purpose table

Information about the available purposes are stored in a table named *purpose table*. A tuple of the purpose table contains the purpose identifier, the purpose name, and the purpose description. Table 4.3 shows the purpose table of doctor Dj.

4.5.4 Trust table

Each peer maintains a local *trust table* which contains the trust level of some peers in the system. A tuple of the trust table contains the identifier of the peer for which is assigned a trust level, the trust level, and a cell defining if this peer is a friend or not. Table 4.4 shows the trust table of doctor Dj.

Private data table j				
Id (PK)	**Data**			**Privacy**
	Table	**Column**	**Id**	**Policy**
PD1	DTj	Birthdate, Gender, Smoker, Diagnoses	—	PP1
PD2	DTj	Country, Birthdate, Gender, Smoker, Diagnosis	—	PP2
PD3	DTj	Diagnosis	Pat3	PP3
PD4	DTj	Diagnosis	Pat5	PP3

Table 4.5: Private data table of doctor Dj

Purpose-based data reference table j			
Purpose	**Operation**	**Data Reference**	**Authorized Requesters**
Purpose1	Read	Pat1	{P25}
		Pat2	{P25,P60}
Purpose2	Write	Pat1	{P31,P27}

Table 4.6: Purpose-based data reference table of doctor Dj

4.5.5 Private data table

Private data table joins the data and the privacy policies tables. Each tuple defines the data subject to privacy (table, column or line) and the corresponding privacy policy id (PPID).

Table 4.5 relies data with privacy policies. This table allows fine-grained access control by specifying which table, column, line, or cell can be accessed by preserving a corresponding privacy policy. For instance, in PD1, only some columns of the data table DTj (those who do not disclose patients identities) are concerned by the privacy policy PP1 where pharmacists and doctors can read records of patients who were born before 2000. In PD3 and PD4 the diagnosis cell of Pat3 and Pat5 can be modified by doctor Dk. It is assumed that doctor Dk is concerned also by PD1 so before modifying the diagnosis she can read the patient record.

4.5.6 Purpose-based data reference table

Information about the references of data allowed for particular purposes and operations are stored in a table named *purpose-based data reference table*. A tuple of this table contains the purpose identifier, the operation, the data reference and the identifiers of requesters allowed to access these data for the specified purpose and operation. Table 4.6 shows the purpose-based data reference table of doctor Dj.

4.6 Functions

In applications based on P2P systems, the basic types of operations a user can do are publishing and requesting data. In this Section, we propose to extend publishing with functionalities that allow the owner to publish private data while preserving their privacy. We also extend data requesting to allow requesters to ask for data while forcing them to respect owners privacy preferences. Finally we propose new functions for purpose-based reference searching for peers who are interested to know which data they can access for a particular purpose.

In the next, we show publishing (cf., Section 4.6.1), requesting (cf., Section 4.6.2) and purpose-based reference searching (cf., Section 4.6.3).

4.6.1 Publishing

In PriMod, we provide users with two ways of publishing or sharing their sensitive data. A user may choose to publish in the system her data content or only data references. In the first case, data privacy is protected from malicious servers by using cryptography techniques. In the second case, there is no need for data encryption since references do not show any private information about the data if they are well-chosen (i.e., do not use personal information such as security numbers, addresses, etc.).

Boolean publishData(data, PPId). Owner peers use this function to publish data content in the system. The second parameter is the privacy policy that dictates the usage conditions and access restrictions of the published data. This function returns true if data content is successfully distributed, false otherwise. This function is similar to a traditional publishing function. To protect data privacy against potential malicious servers, before distribution, data content is encrypted (symmetric cryptography) (cf., Section 5.3.2.4), and digital checksums are used to verify data integrity. Symmetric keys are stored locally by the owner. Requesters must contact owners to retrieve keys and decrypt requested data.

Boolean publishReference(data, PPId). Owner peers use this function to publish data references in the system while data content are stored locally. The second parameter is the privacy policy that dictates the usage conditions and access restrictions of the published data references. This function returns true if data references are successfully distributed, false otherwise. Servers store data references and help requesters to find data owners to obtain data content. Publishing only data references allows owners to share private data while being sure that data content will be provided to right requesters. This hypothesis can not be guaranteed in the previous function because servers may misbehave by returning encrypted data to unauthorized peers.

4.6.2 Requesting

Requesters must not be concerned by the way data are published (data or references). Requesting must be done in a transparent way, thus a unique requesting function is proposed in PriMod.

Data request(dataRef, purpose, operation). Requester peers use this function to request data (*dataRef*) for a specific purpose (e.g., researching, diagnosis, analysis) to perform a specific operation (i.e., read, write, disclose). This function returns the requested data if the requester has corresponding rights, otherwise it returns null. This function compels requesters to specify the access purposes and the operations that they will apply to requested data.

This explicit request is the cornerstone of this work, it commits requesters to use data only for the specified purposes and to perform only specified operations. Legally, this commitment may be used against malicious requesters if it is turned out that obtained data have been used differently.

TrustL searchTrustL(requesterID, NestedLevel). Owner peers use this function to search the trust level of the requester *requesterID*. *NestedLevel* defines the level of nested searching (e.g., 0 for the owner, 1 for the first contacted peers, etc.). This function returns the trust level of the requester if it is found else it returns 0. This trust level is used in the requesting process to verify the trustworthiness of the requester in order to give him access rights.

4.6.3 Purpose-based reference searching

Peers may be interested in finding which data they are allowed to ask for a particular purpose. PriMod provides users with a function for purpose-based reference searching. Purpose-based reference publishing can be done as follows. The allowed data reference lists can be created transparently while publishing data. These lists can be published periodically in the system, or on the demand of a third-party (e.g., health care department).

Purpose-based reference searching allows requesters to know which data they are authorized to request or use which prevents them from denying their knowledge about their access rights. It also allows peers to find data references while avoiding a global schema. A global schema is complex to achieve in distributed environments and may have a negative impact on data privacy. Global information about data may contain sensitive elements which must not be disclosed to all peers.

DataRefList dataRefSearch(purpose, operation). Requester peers use this function to request data references (*dataRefList*) of data they are authorized to access for a specific purpose and operation. This function returns the requested data reference list if it exists, otherwise it returns null.

4.7 Conclusion

This chapter has shown PriMod a privacy model for P2P systems that includes purpose-based access control, trust and cryptography to protect data based on the owners privacy preferences. PriMod makes no assumptions about the type of P2P system and the type of data. Thus it can be used in many P2P applications where sensitive data are shared such as medical or research collaborative networks.

To resume, PriMod propose a privacy policy model that allows data owners to specify their privacy preferences, a trust model in order to control the trustworthiness of requesting peers, and a data model that relies data to privacy policies. It allows two types of publishing, an owner can choose to publish encrypted data or data references. Requesting data is always done for a particular purpose and operation which commits requesters to use data only for the specified purposes and operations. We also propose a purpose-based reference searching which allows requesters to find references of data they can access for a particular purpose and operation.

PriMod	Used privacy techniques	Access control techniques		Use of purpose-based access control
		Trust techniques		Use of trust levels
		Data encryption		Symmetric encryption
		Data integrity protection		Use of content hashing as a checksum
	Guaranteed properties	Protection against unauthorized reads		Yes due to encryption
		Protection against corruption and deletion		Yes due to data checksum
		Limited disclosure	Owner	Yes due to access control
			Requester	Yes due to access control
		Content deniability		Yes due to encryption

Table 4.7: PriMod: used privacy techniques and guaranteed properties

By comparing PriMod to the data privacy models shown in Chapter 2:

- As many models, private data in PriMod are protected against malicious reads, corruptions and deletions by using encryption and data checksums. Data disclosure is limited by using access control techniques.
- PriMod is used to prevent data misuses. It needs to be extended with auditing functionalities in order to verify the correct use of data after access.
- PriMod does not guarantee anonymity since it is not designed to protect users behaviors but focuses on data privacy protection.
- Unlike all the models presented, PriMod includes the notion of purpose by implying Hippocratic principles to the data model.

Next chapter shows how PriMod is used as a base to implement PriServ, a privacy service for DHT-based P2P systems.

Chapter 5

PriServ, a privacy service for DHT-based P2P systems

This chapter presents PriServ, a privacy service for P2P systems based on Distributed Hash Tables (DHT) that implements PriMod. To our knowledge, PriServ is the first proposition that introduces purpose-based access control in P2P systems.

5.1 Introduction

PriServ is a privacy service on top of DHTs which, based on PriMod, prevents privacy violation by limiting malicious data access. For that, we use purpose and operation based access control, trust techniques, cryptography techniques and digital checksums.

PriServ only uses the get() and put() functions of the DHT, and proposes to users an interface for publishing and requesting private data securely. Owners are given the capacity to share their sensitive data while protecting their data privacy.

This chapter is organized as follows. Section 5.2 introduces the design choices of PriServ. Section 5.3 presents the PriServ architecture. Section 5.4 illustrates the algorithms of PriServ by using the motivating example. Section 5.5 concludes.

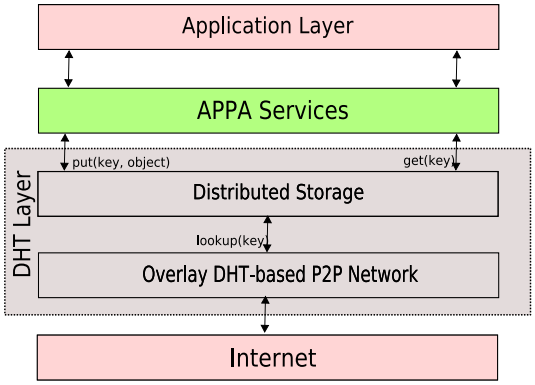


Figure 5.1: Global architecture

5.2 Design choices

PriServ uses the get() and put() functions of the DHT to localize and publish data. Data keys, which are created by using the hash functions, contain the notions of purpose and operation. More notions can be added to data keys, however we assume that these information are sufficient to identify the requester intentions on the use of the data. The following discusses the use of DHT functions (cf., Section 5.2.1) and data keys generation (cf., Section 5.2.2).

5.2.1 Use of DHT functions

As said in Section 3.2.2, in all DHT systems (e.g., Chord [SMK+01], Pastry [RD01], etc.), data location is based on associating a *key* with each data item, and storing the key/data item pair at the peer to which the key maps.

- $put(k, object)$ stores a key k and its associated data object in the DHT.
- $get(k)$ retrieves the data object associated with k in the DHT.

Figure 5.1 shows the considered global architecture. On top of the Internet network there is the P2P system. The overlay network layer takes in charge the routing mechanism by implementing the *lookup()* function but also by managing peers dynamicity (join/leave of peers). On top of this layer, the distributed storage layer ensures key-based data searching and data distribution by implementing the put() and get() functions. Those two layers make abstraction of the DHT-based P2P system and in the next we will refer to them as the DHT.

PriServ is implemented as an APPA service [AMPV06] on top of the DHT. The PriServ implementation uses Chord for its efficiency and simplicity. Nevertheless, all DHT-based P2P system can be used.

In Chord, a DHT maps a key k to a peer P called *responsible for k with respect to a hash function h*. Peers maintain information about $O(logN)$ other peers in a *finger table* and resolve lookups via ***O(logN)*** messages to other peers. A finger table entry includes both the Chord identifier and the IP address (and port number) of the relevant peer. A consistent hash function assigns to each peer and key an m-bit *identifier* using a base hash function such as SHA-1 [SHA95]. A peer identifier is chosen by hashing the peer IP address. A key identifier is based on data values that can be a data reference, an address, etc. All peers are ordered in a circle modulo 2^m. Key k is assigned to the first peer whose identifier is equal to or follows k in the identifier space. This peer is called the *successor* of k. Chord provides data publishing and searching using only ***O(logN)*** messages.

5.2.2 Data keys generation

To simplify, consider that the DHT generates peer identifiers by hashing the peer IP address (like in Chord). Concerning data keys, as seen in PriMod operations (cf., Section 4.6) publishing and requesting are always done for a particular privacy policy (PPID), in particular, a purpose and an operation, thus we propose to hash the triplet (*dataRef, purpose, operation*) to create data keys. *dataRef* is a unique data reference, *purpose* is the data access purpose and *operation* is the operation that can be executed on requested data with respect to the corresponding privacy policy (PPID). Thus, the same data with different access purposes and different operations have different keys[1].

Because keys contain the notion of purposes and operations, requesting data is always made for a defined purpose and operation. This allows to enhance access control because to construct data keys, data requesters have to include the purpose and operation for which data are accessed.

Data references are expressed as follows. Inspired by PIER [HHL+03], we propose to concatenate the data table name, the column name and the value of the primary key. By using this last value, the searching granularity can be a tuple. If shared data concerns an entire table, the column name and the value of the primary key are omitted. Similarly, if a particular column is concerned, the value of the primary key is omitted. If several columns of the same table are concerned they can be enclosed in curly brackets. For instance, in table 4.5, the data reference for PD1 is DTj.{Birthdate, Gender, Smoker, Diagnoses} and for PD3 is DTj.Diagnosis.Pat3.

5.3 PriServ architecture

Figure 5.2 shows the PriServ architecture. The PriServ layer is a middle layer between the application layer and the DHT. It offers four interfaces to the application layer, which

[1] [Fur05] has analyzed how to manage schemas and process simple queries efficiently in a flexible P2P environment. Thus, we consider that all peers accessing the same data are capable of schema mapping and that peers allowed to access particular data are informed of their allowed purposes. Thus, requester peers are able to produce keys.

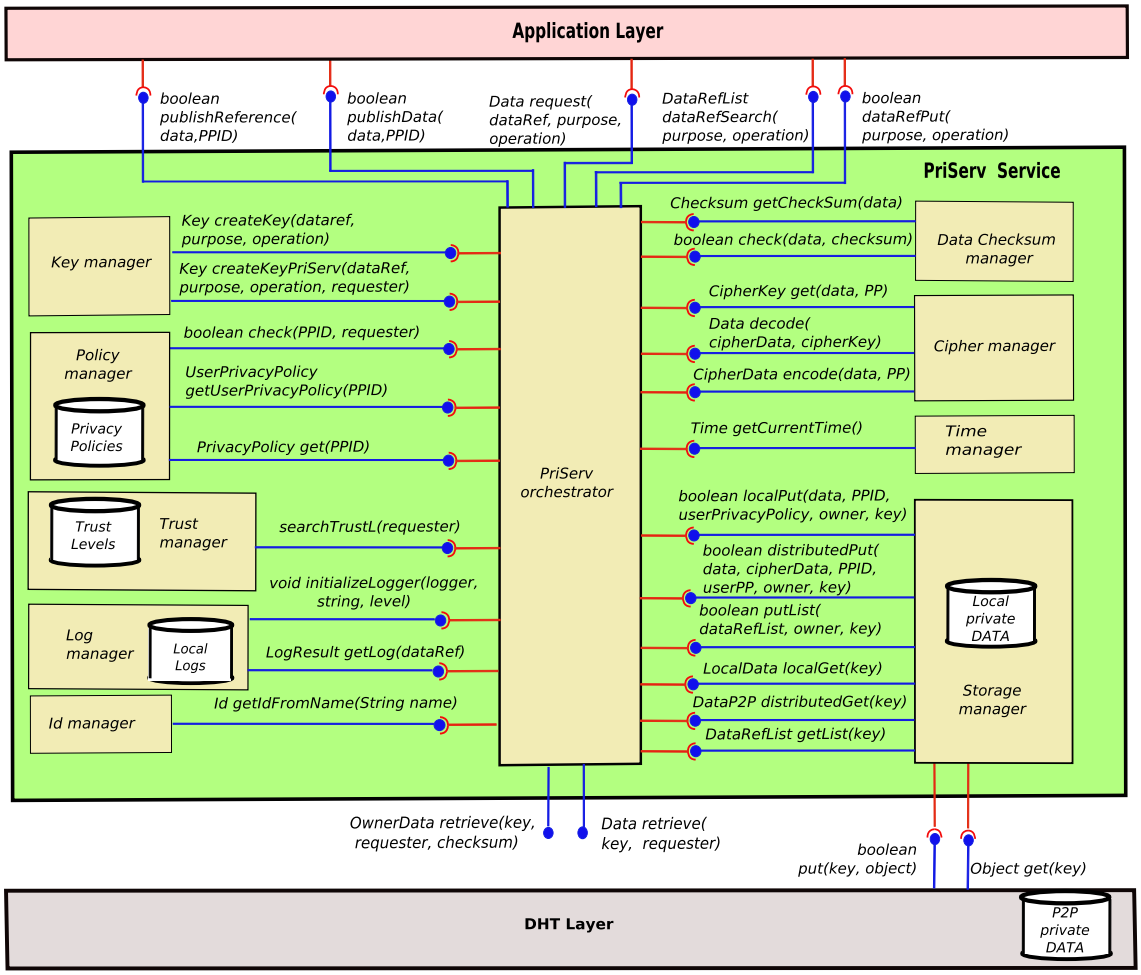


Figure 5.2: PriServ architecture

correspond to the four PriMod operations (publishReference, publishData, request, and dataRefSearching) (cf., Section 4.6). It also offers two interfaces (retrieve functions) which are used for the interaction between requesters and owners. PriServ uses the two interfaces of the DHT layer (put and get).

In the PriServ layer, we find several components gathered around an orchestrator which organizes PriServ activities. Each component is responsible of a PriServ activity. The orchestrator may execute three different workflows depending on the peer role. In the following, we present each of those components (cf., Section 5.3.1) and the PriServ algorithms (cf., Section 5.3.2).

5.3.1 Components

PriServ is composed of independent managers which are responsible of its main activities. We choose these activities based on two criteria:

- To provide a P2P functionality. Based on this criterion, we propose a storage manager, responsible of data storage, a key manager responsible of data hash keys used in the DHT, a time manager responsible of time synchronization, and an id manager responsible of peers' identifiers.
- To guarantee a privacy property. Based on this criterion, we propose a policy manager, a trust manager, a cipher manager, a data checksum manager and a log manager which are respectively responsible for privacy policy management, trust, encryption, checksum management and logging.

Other managers can be added in order to extend PriServ activities. In the following, we present the managers shown in Figure 5.2.

5.3.1.1 Storage manager

Its role is to manage data storage. Stored data are those of Tables 4.1 and 4.5. Data storage is made locally before data are stored in the P2P system[2]. To store data in the P2P system, this component invokes the put() and get() functions of the DHT layer.

The operations provided by the storage manager are:

- `boolean localPut(Data data, PrivacyPolicyID PPID, UserPrivacyPolicy userPP, String owner, Key key)` creates and stores the owner identifier on the DHT. This operation is invoked by the publishing references function. It also stores a copy of the data locally.
- `boolean distributedPut(Data data, CipherData cipherData, PrivacyPolicyID PPID, UserPrivacyPolicy userPP, String owner, Key key)` creates and stores the encrypted data on the DHT. This operation is invoked by the publishData() function. It also stores a copy of the data locally.
- `boolean putList(DataRef dataRef, String Owner, Key key)` stores the data reference list allowed for a particular purpose on the DHT.
- `LocalData localGet(Key key)` retrieves the data stored locally corresponding to key.
- `DataP2P distributedGet(Key key)` sends a query to request data corresponding to key. It invokes the get function of the DHT then the retrieve function.
- `DataRefList getList(Key key)` sends a query to request the data reference list corresponding to key.

[2]This allows to keep an original replica of the data in case of a malicious data modification. Eventually, this replica can be optional.

5.3.1.2 Policy manager

Its role is to manage PPs. Data owners organize their privacy preferences in PPs (cf., Figure 4.3) which are stored by the policy manager in Table 4.2. From PPs, this component creates user privacy policies that should be respected by requesters. From PP, the component can also extract the list of authorized users. This list is a kind of ACL (Access Control List) that contains only allowed users. We recall that in this work, to simplify, we consider that users may belong to groups. Thus, this ACL contains mainly a list of groups of users and maybe some individual users known in advance.

For the time being, in the implementation, we only consider the purpose, the operation and the ACL in the user privacy policies. Other concepts could be added later.

The operations provided by the policy manager are:

- `PrivacyPolicy get(PrivacyPolicyID PPID)` returns the privacy policy corresponding to PPID.
- `UserPrivacyPolicy getUserPrivacyPolicy(PrivacyPolicyID PPID)` returns a short version of the privacy policy corresponding to PPID. This privacy policy is published with the data.
- `boolean check(PrivacyPolicyID PPID, String requester)` verifies the access rights of the requester for the privacy policy PPID.

5.3.1.3 Trust manager

Its role is to manage trust levels. We consider that each peer stores locally a list of peers and the corresponding trust levels. If a required trust level does not exist locally, the trust manager asks for it to other peers. In PriServ three ways of obtaining trust levels are used, namely, with-friends, without-friends and with-or-without-friends (cf., Section 5.3.2.2). The use of the trust manager is optional because it may generate high costs.

The operation provided by the trust manager is:

- `TrustLevel searchTrustL(String requester)` searches for the requester trust level.

5.3.1.4 Cipher manager

Its role is to manage cryptography in PriServ. It offers a function that creates a symmetric cipher key for each pair (data, PP). It also offers two functions to encrypt and decrypt data. In this work, symmetric-key algorithms can be used because they are generally much less computationally intensive than other cryptography algorithms (cf., Section 2.6.1). However, PriServ is independent of the encryption techniques used to encrypt data.

Concerning symmetric-key transmission, maybe the unique completely secure way of transmitting keys is from hand to hand between individuals. In this work, we consider that keys can be transmitted by using public-key cryptography. To do so, when requesters contact data owners to retrieve data, they send their public key. Thus, owners will encrypt

the data encryption key with the requester's public key. In this way, only the allowed requester will be able to decrypt the encryption key with its private key.

The operations provided by the cipher manager are:

- `CipherKey get(Data data, PrivacyPolicy PP)` returns the secret key corresponding to data and PP.
- `Data decode(CipherData cipherData, CipherKey cipherKey)` decrypts cipherData with the secret key cipherKey.
- `CipherData encode(Data data, PrivacyPolicy PP)` encrypts data with the secret key corresponding to PP.

5.3.1.5 Data checksum manager

Its role is to check the integrity of data. This component offers a function to calculate digital data checksums (e.g., by using MD5). Digital checksums can be used by owners to verify if servers have tampered with the data. PriServ is independent of the techniques used to create data checksums.

The operations provided by the data checksum manager are:

- `Checksum getCheckSum(Object data)` creates a digital checksum from the data using a hashing algorithms (e.g., MD5).
- `boolean check(Object Data, Checksum checksum)` checks if a checksum corresponds to a datum.

5.3.1.6 Key manager

Its role is to generate data keys. It creates data keys by hashing data references, purposes and operations. This component offers two functions. The first one, used in publishReference() and publishData(), returns the created data key. The second, used in request(), returns the created data key and appends to it the requester identifier in order to identify the requester during the requesting process.

The operations provided by the key manager are:

- `Key createKey(DataRef dataref, Purpose purpose, Operation operation)` creates key by hashing a data reference, a purpose and an operation. This key is used for data publishing.
- `Key createKeyPriServ(DataRef dataRef, Purpose purpose, Operation operation, String peerId)` creates a key and append to it the peerID. This key is used for data requesting.

5.3.1.7 Id manager

Its role is to identify peers from their names (e.g., URI). Peer identifiers are used in access control to authorize or prohibit access to data.

The operation provided by the id manager is:

- `Id getIdFromName(String name)` returns the peer identity corresponding to its name.

5.3.1.8 Log manager

Its role is to manage logs. It stores logs in a dedicated database on each peer and retrieves them when auditing the system.

The operations provided by the log manager are:

- `void initializeLogger(Logger logger, String string, Level level)` initializes the logger. This includes the creation of corresponding relational tables.
- `LogResult getLog(DataRef dataref)` retrieves the logs corresponding to data.

5.3.1.9 Time Manager

Its role is to give the current time. It is used for synchronizing clocks of all connected peers.

The operation provided by the time manager is:

- `public String getCurrentTime()` gives the current time modulated by an offset to have a synchronized clock between all peers.

5.3.1.10 PriServ orchestrator

It is the central component of PriServ. According to the peer' role, the orchestrator executes a different workflow by using the components introduced before.

- ***Owner orchestrator.*** Its role is to orchestrate the owner functionalities. It is responsible of publishing owner references or data depending on the called function (publishData() or publishReference()). It is also responsible of retrieving data or symmetric keys during the requesting process. It interacts with the application layer for publishing and with the requester orchestrator for retrieving.
- ***Requester orchestrator.*** Its role is to orchestrate the requester data requesting. It interacts with the application layer for requesting and with the owner orchestrator for retrieving.
- ***Server orchestrator.*** Its role is to orchestrate the server functionalities. For that, it interacts with the DHT layer to store data of the P2P system and to return stored data.

The following section presents the operations provided by the orchestrators.

5.3.2 Functions

This section presents the PriServ algorithms: purpose-based reference searching algorithms, three ways of trust level searching (with-friends, without-friends and with-or-without-friends), two ways of publishing (publishReference() and publishData()) and two steps for requesting (data requesting to the server and data retrieving from the owner).

5.3.2.1 Purpose-based reference searching

PriServ implements the purpose-based reference searching function of PriMod. As mentioned in Section 4.6.3, purpose-based reference lists are created and updated while publishing data. For that, The following lines are added to the publishing reference algorithm (line 6 and 7 of Figure 5.6) and the publishing data algorithm (line 7 and 8 of Figure 5.7) :

```
Owner orchestrator
6 (resp. 7): key2 = keyManager.createKey(null, privacyPolicy.purpose,
                                          privacyPolicy.operation);
7 (resp. 8): storerManager.addRef(key2, data.getDataref(), privacyPolicy.ACL);
```

Each time an owner publish data (or data references), the owner orchestrator asks the key manager to create a second key by hashing only the purpose and the operation. Then it asks the storage manager to add the data reference and its correspondent ACL in a list called data reference list.

addRef(key, dataRef, ACL). When the storage manager of the owner receives this call, it uses the algorithm shown in Figure 5.3. In line 3, the storage manager checks if its purpose-based data reference table contains an entry *KeyListRef* which is a list of data references corresponding to the purpose hash key:

- If *KeyListRef* does not exist, it creates a new one in line 4. For each requester contained in the ACL, it creates a list *dataRefList* of allowed data references and adds to it *dataRef* in line 6 and 7. Then it adds a new couple composed by the requester identifier and *dataRefList* to *KeyListRef* in line 8.

- If *KeyListRef* exists, for each requester contained in the ACL, it searches for its list *dataRefList* of allowed data references in line 12. If *dataRefList* does not exist, it creates a new one in line 14. Then, it adds *dataRef* to *dataRefList* in line 16. Finally, it adds the couple composed by the requester identifier and the updated *dataRefList* to *KeyListRef* in line 17.

```
Owner storage manager
0:   addRef(key, dataRef, ACL)
1:   begin
2:     KeyListRef = localDataRefTable.get(key);
3:     if (KeyListRef does not exist)
4:       create new KeyListRef;
5:       for each authorized requesteri contained in the ACL
6:         create new dataRefList;
7:         dataRefList.add(dataRef);
8:         KeyListRef.add(ACL.get(i), dataRefList);
9:       endfor
10:    else
11:      for each authorized requester i contained in the ACL
12:        dataRefList = KeyListRef.get(ACL.get(i));
13:        if ( dataRefList does not exist)
14:          create new dataRefList;
15:        endif;
16:        dataRefList.add(dataRef);
17:        KeyListRef.add(ACL.get(i), dataRefList);
18:      endfor;
19:    endif;
20:  end;
```

Figure 5.3: addRef() function

```
Requester orchestrator
0:   DataRefList dataRefSearch(purpose, operation)
1:   begin
2:     dataRefList = null;
3:     key = keyManager.createKeyPriServ(null, purpose, operation, requesterID);
4:     dataRefList = storageManager.GetList(key);
5:     return dataRefList;
6:  end;

Requester storage manager
7:   DataRefList GetList(key)
8:   begin
9:     dataRefList = DHT.get(key);
10:    return dataRefList;
11:   end;
```

Figure 5.4: dataRefSearch() function

The publication of the data references lists can be done periodically in the DHT by using the put function. The publication can also be initiated by a third-party (e.g., health care department on the demand of a scientist).

dataRefSearch(purpose, operation). In PriServ, requesters are able to know the references of data they can access for a particular purpose and operation. When the requester orchestrator receives this call, it uses the algorithm shown in Figure 5.4. In line 3, the requester orchestrator asks the key manager to create a key by hashing the purpose and the operation. Then, it asks its storage manager to get the data reference list from the P2P system in line 4. The storage manager contacts the server storing the data reference list corresponding to the created hash key by invoking the get() function of the DHT.

When a server peer receives a get message, it searches in its table an entry corresponding to the key and verify for the requester identifier if a data reference list exists. Then it returns the stored data reference list to the requester storage manager in line 9. Finally, the requester storage manager returns the data reference list to its orchestrator in line 10.

5.3.2.2 searchTrustL(requesterID, NestedLevel)

If the owner trust manager has the trust level of the requester in its trust table (Figure 5.5 line 10), this level is returned directly and the owner does not have to contact other peers. Otherwise, PriServ defines three methods for searching the requester trust level. Choosing one of them depends on one of three hypotheses.

- Each peer have at least one friend (with-friends algorithm).
- Each peer does not have any friends (without-friends algorithm).
- Each peer may have friends or not (with-or-without-friends algorithm).

With-friends. (Figure 5.5). By making this assumption, the owner asks its friends for the trust level of the requester (line 16). Each received trust level (RTL) is weighted with the trust level (FTL) of the sending friend (line 18). The final trust level is computed from the received trust levels (line 20) and the computational function can be the average, the maximum, the minimum, etc. This searching is recursive. If a friend does not have the requested trust level it asks for it to its friends and the number of nested levels (*NestedLevel*) is incremented (line 12). Recursion is limited by a predefined number of iterations (MaxDepth, line 3). The maximum number of contacted friends can also be limited to a predefined number.

Without-friends. In this case, an owner asks peers contained in its finger table, which are easy to contact, for the trust level of the requester. This is done by substituting lines 14 to 19 in Figure 5.5 by these lines:

```
0: searchTrustLevel(requesterID,nestL)
1: begin
2:  requesterTrustLevel is initialized to 0;
3:  if(nestL has reached MaxDepth)
4:   if(trustL of requesterID in trustTable)
5:    requesterTrustL=trustL of requesterID;
6:   else
7:    return -1;
8:  else
9:  if(trustL of requesterID in trustTable)
10:   requesterTrustL=trustL of requesterID;
11: else
12:   nestL is incremented;
13:   NbPeersContacted is initialized to 0;
14:   for each friend
15:     FTL = trustL of friendID;
16:     RTL = friendTrustManager.searchTrustLevel(requesterID,nestL);
17:     if (RTL != -1)
18:      FTL*RTL is added to requesterTrustL;
19:      NbPeersContacted is incremented;
20:   requesterTrustL = requesterTrustL / NbPeerscontacted;
21: return requesterTrustL
22:end;
```

Figure 5.5: Trust function: with-friends

```
14: for each peer contained in the finger table
15:   RTL = PeerTrustManager.searchTrustLevel(requesterID, NestedLevel);
16:   RTL is added to requesterTrustL;
17:   NbPeersContacted is incremented;
```

The final trust level is computed from the received trust levels and the computational function can be the average, the maximum, the minimum, etc. This searching is recursive. Recursion is limited by a predefined number of iterations (MaxDepth).

With-or-without-friends. In this case, priority in asking for trust levels is given to friends. If an owner has some friends, it asks them for the trust level by using the with-friends algorithm, else the owner asks the peers in its finger table by using the without-friends algorithm. The final trust level is computed from the received trust levels and the computational function can be the average, the maximum, the minimum, etc. This searching is recursive. Recursion is limited by a predefined number of iterations (MaxDepth).

```
Owner orchestrator
0:   publishReference(data, PPID)
1:   begin
2:     privacyPolicy = policyManager.get(PPID);
3:     userPP = policyManager.getUserPrivacyPolicy(PPID);
4:     key = keyManager.createKey(data.dataref, privacyPolicy.purpose,
                                  privacyPolicy.operation);
5:     storageManager.localPut(data, PPID, userPP, privacyPolicy.ACL,
                                owner, key);
6:     key2 = keyManager.createKey(null, privacyPolicy.purpose,
                                   privacyPolicy.operation);
7:     storerManager.addRef(key2, data.getDataref(), privacyPolicy.ACL);
8:   end;

Owner storage manager
9:   localPut(data, PPID, userPP, ACL, owner, key)
10:   begin
11:     DataTable.localSave(data);
12:     PrivateDataTable.localSave(data.dataRef,PPID);
13:     dataP2P = createDataP2P(userPP, ACL, owner);
14:     DHT.put(key,dataP2P);
15:   end;
```

Figure 5.6: publishReference() function

5.3.2.3 publishReference(data, PPID)

When the owner orchestrator receives this call, it uses the algorithm shown in Figure 5.6 to publish data references in the P2P system and to store data locally. In line 0, the owner application sends the privacy policy identifier (PPID) and the data to publish. In lines 2 and 3, the owner orchestrator asks the policy manager for the PP and the user privacy policies of PPID. Then, it asks the key manager to create the data key in line 4. Finally, the orchestrator asks the storage manager to locally put the data in line 5.

In lines 11 and 12, the storage manager stores locally the data in the data table (Table 4.1), then the data reference and PPID in the private data table (Table 4.5). In line 13, it creates the P2P data that will be stored on the P2P system which is composed of the owner identifier, the user privacy policy and the ACL. Finally, in 14, it invokes the DHT to put the P2P data on the P2P system.

5.3.2.4 publishData(data, PPID)

When the owner orchestrator receives this call it uses the algorithm shown in Figure 5.7 to publish the data content in the P2P system and to store locally a copy of the data. Lines 2 to 4 are the same as in the publishReference() function (same lines). In order to put data on the P2P system, data should be encrypted, the orchestrator contacts the cipher manager to obtain the cipher data in line 5. Finally, in line 6, the orchestrator asks the storage manager to store locally the data then to put them on the P2P system. Notice that in this step, cipher data are transmitted to the storage manager.

```
Owner orchestrator
0:   publishData(data, PPID)
1:   begin
2:     privacyPolicy = policyManager.get(PPID);
3:     userPP = policyManager.getUserPrivacyPolicy(PPID);
4:     key = keyManager.createKey(data.dataref, privacyPolicy.purpose,
                                  privacyPolicy.operation);
5:     cipherData = cipherManager.encode(data, privacyPolicy);
6:     storageManager.distributedPut(data , PPID, cipherData,
                                     userPP, privacyPolicy.ACL, owner, key);
7:     key2 = keyManager.createKey(null, privacyPolicy.purpose,
                                   privacyPolicy.operation);
8:     storerManager.addRef(key2, data.getDataref(), privacyPolicy.ACL);
9:   end;

Owner storage Manager
10:   distributedPut(data , PPID, cipherData, userPP, ACL, owner, key);
11:   begin
12:     DataTable.localSave(data);
13:     PrivateDataTable.localSave(data.dataRef,PPID);
14:     dataP2P = createDataP2P(cipherData, userPP, ACL, owner);
15:     DHT.put(key,dataP2P);
16:   end;
```

Figure 5.7: publishData() function

Lines 12, 13 executed by the storage manager are the same as in the publishReference() function (lines 11, 12). Then, it creates, in line 14, the P2P data to store on the P2P system which is composed of the cipher data, the owner identifier, the user privacy policy and the ACL. Finally the storage manager invokes the DHT to put the P2P data on the P2P system.

It is important to underline that the DHT is used in the same way in both publishing functions, lines 14 in the first function and 15 in the second one. Only the content of the data varies since sent data are constructed differently.

5.3.2.5 Request(dataRef, purpose, operation)

When a requester orchestrator receives this call, it uses the algorithm shown in Figure 5.8. In line 3, the requester orchestrator asks the key manager to create the data key. Then, it asks its storage manager to get the data from the P2P system in line 4. The storage manager contacts the server storing the data corresponding to the created data key by invoking the get() function of the DHT.

When a server peer receives a get message it checks if the requester is authorized to access the data by using the ACL attached to each data. Then it returns the stored P2P data to the requester storage manager in line 22. Finally, the storage manager returns the P2P data to its orchestrator in line 23.

The P2P data contains either cipher data (line 5) or a data reference (line 15).

```
Requester orchestrator
0:   Data request(dataRef, purpose, operation)
1:   begin
2:     data = null;
3:     key = keyManager.createKeyPriServ(dataRef, purpose, operation, requester);
4:     dataP2P = storageManager.distributedGet(key);
5:     if (dataP2P contains cipher data) do
6:        checksum = dataCheckSumManager.getCheckSum(dataP2P.cipherData);
7:        ownerData = dataP2P.owner.retrieve(key, requester, checksum);
8:        if (ownerData contains a cipherKey) do
9:           data = cipherManager.decode(dataP2P.cipherData,ownerData.cipherKey);
10:       else if (ownerData contains data) do
11:          data = ownerData.data;
12:          endIf;
13:       endIf;
14:    endIf;
15:    if (dataP2P contains a data reference) do
16:       data = dataP2P.Owner.retrieve(key, requester);
17:    endIf;
18:    return data;
19:  end;

Requester storage manager
20:   dataP2P distributedGet(key)
21:   begin
22:     dataP2P = DHT.get(key);
23:     return dataP2P;
24:   end;
```

Figure 5.8: request() function

- In the first case, the requester orchestrator asks to its data checksum manager the checksum of the received cipher data in line 6. In line 7, it asks the owner orchestrator the symmetric cipher key. The owner orchestrator sends the cipher key or the data (unencrypted) if its data checksum differs from the requester' one. Then, the requester orchestrator asks the cipher manager to decrypt the cipher data with the symmetric key in line 9.
- In the second case, in line 16, the requester orchestrator asks the owner orchestrator the data.

Finally, the orchestrator returns the requested data to the application (line 18).

5.3.2.6 Retrieve(key, requester, checksum)

The owner orchestrator uses the retrieve function to answer the requester when it asks for data or symmetric keys. Figure 5.9 shows the retrieving algorithm. In line 2, the owner orchestrator asks its storage manager to get locally the data corresponding to the received key. Then, in line 3, it asks its policy manager to check if the requester has the right to access data. If he has it, the orchestrator asks the trust manager to find the requester

```
Owner orchestrator
0:   OwnerData retrieve(key, requester, checksum)
1:   begin
2:     localData = storageManager.localGet(key);
3:     if (policyManager.check(localData.PPID,requester))
4:        requesterTrustL = trustManager.searchTrustLevel (requester, 0);
5:        privacyPolicy = policyManager.get(localData.PPID());
6:        if( requesterTrustL >= privacyPolicy.minTrustL )
7:          localChecksum = dataCheckSumManager.getCheckSum(localData.cipherData);
8:          if (localChecksum.equals(checksum)) do
9:             cipherKey = cipherManager.get(localData.data,privacyPolicy);
10:            return OwnerData(cipherKey);
11:         endIf;
12:         return OwnerData(localData.data);
13:       endif;
14:    endIf;
15:    return null;
16:  end;
```

Figure 5.9: retrieve() function

trust level in line 4. If the trust level is higher or equal to the trust level specified in the corresponding PP, in line 8, the owner orchestrator asks the data checksum manager to check if the requester data checksum differs from the local checksum of the data. In this case, the requester has a valid cipher data and the owner sends him the symmetric cipher key in line 10. If the data are damaged, the owner sends to the requester the corresponding data in line 12. If data are damaged, there is a probability that the server has attacked the integrity of the data. This means that it violates data privacy and this cheating should be punished, for instance by lowering the trust level of the server.

5.4 Illustrating examples

This section illustrates PriServ algorithms on many examples. In the following, consider 5 peers with identifiers P23[3], P31, P33, P51 and P60. Consider also that:

- P23 is an owner peer.
- P31 is a requester peer.
- P33, P51 and P60 are server peers.

5.4.1 Publishing

Figure 5.10 illustrates the publishing reference algorithm. P23 shares data with keys 34, 40 and 58. $hash1$ is used to produce keys from data identifiers (IDri), purposes and

[3]To distinguish data keys from peer keys, we prefix peer keys with letter P.

operations. P51 is the server peer responsible for key 34 and 40 and P60 for 58. Only references of P23 data are published with its identifier by using the put function.

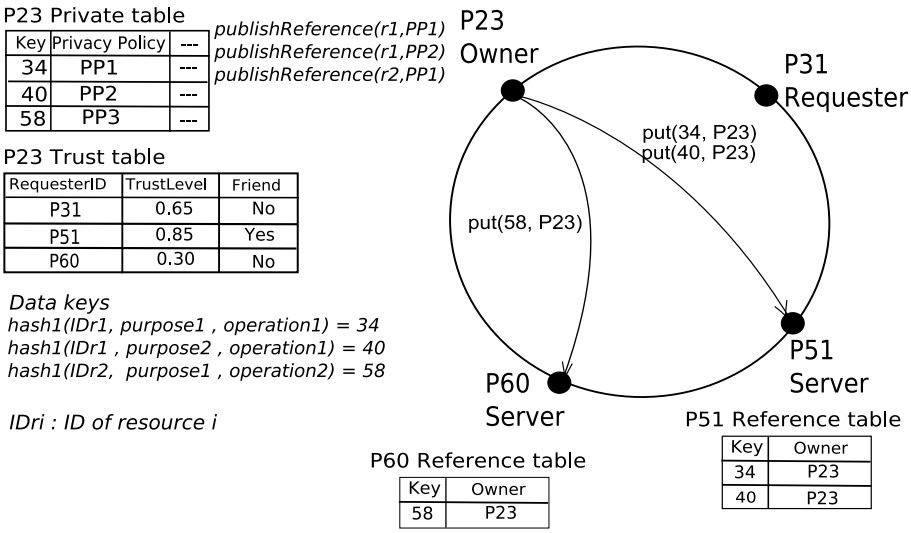


Figure 5.10: Reference publishing example

Figure 5.11 illustrates the publishing data algorithm. P23 shares data with keys 36, 42 and 59. *hash*1 is used to produce keys from data identifiers (IDri), purposes and operations. P51 is the server peer responsible for key 36 and 42 and P60 for 59. P2P data are created by P23 from encrypted data, P23 identifier and an ACL which contains requesters' identifiers allowed to access the concerned datum. P2P data are published by using the put function.

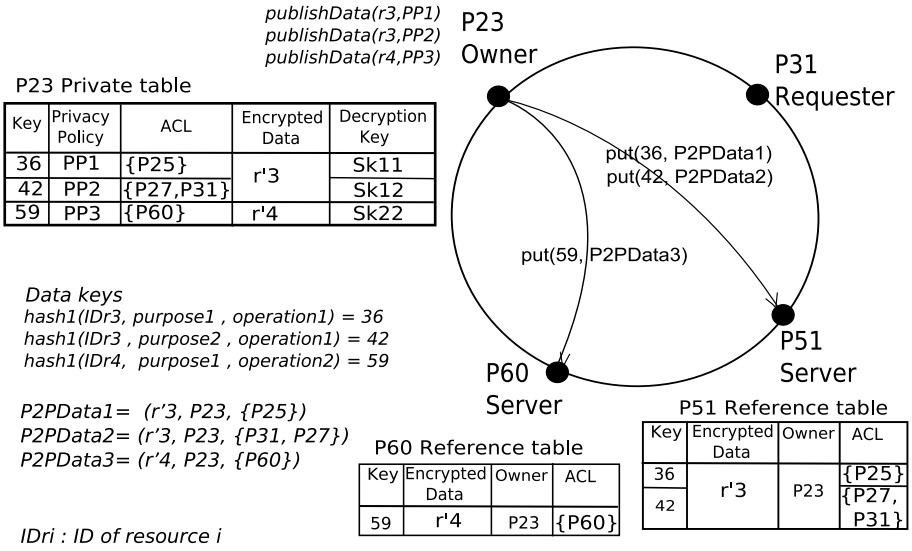


Figure 5.11: Data publishing example

5.4.2 Requesting

Figure 5.12 illustrates the data requesting algorithm on the example where data references are published (cf., Figure 5.10). P31 requests data reference IDr1 for purpose2 and operation1 which correspond to key 40 ($get(40)$). The peer which identifier is equal to or follows 40 is P51. P51 returns P23 which is the owner peer of 40. Then P31 contacts P23 to obtain the data corresponding to 40 ($retrieve(40)$). P23 verifies the information contained in the privacy policy of 40 (PP2). In this example, the trust table of P23 contains the trust level of P31. As it is higher than the level required in PP2 (i.e., 0.6)(cf., Table 4.2) the access is granted.

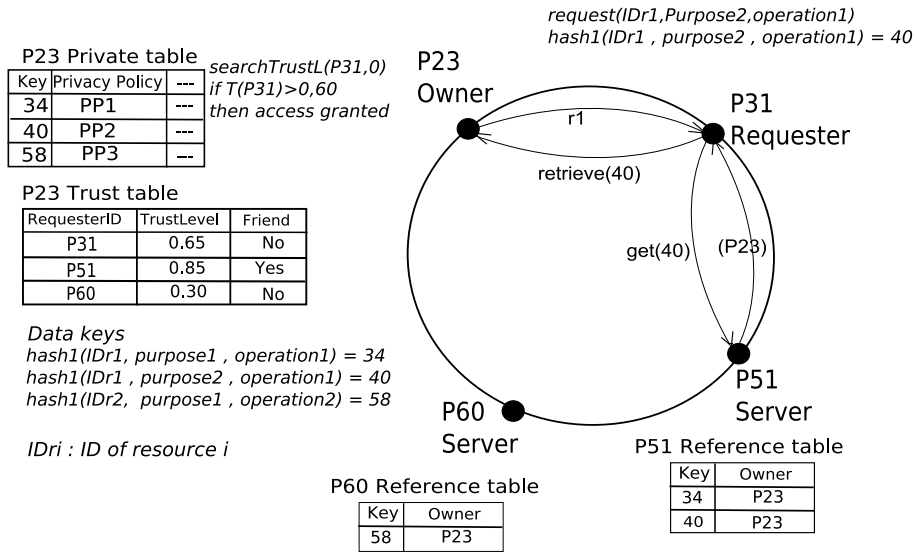


Figure 5.12: Data requesting example when references are published

Figure 5.13 illustrates the data requesting algorithm on the example where encrypted data are published (cf, Figure 5.11). P31 requests data for purpose2 and operation1 which corresponds to key 42 ($get(42)$). The peer which identifier is equal to or follows 42 is P51. P51 returns P23 which is the owner peer of 42 and the encrypted data corresponding to 42. In this example we consider that P51 misbehaves and returns a corrupted data $cr'3$. P31 calculates a checksum of the received data then it contacts P23 to obtain the decryption key corresponding to 42 ($getSymKey(P23, checksum, 42)$). P23 verifies the information contained in the privacy policy of 42 (PP2). In this example, the trust table of P23 contains the trust level of P31. As it is higher than the level required in PP2 (i.e., 0.6)(cf., Table 4.2) the access is granted. P23 calculates the checksum of the data corresponding to 42 and compares it to the checksum value sent by P31. P23 finds out that the checksums are not equal and deduces that someone has misbehaved. Since P31 has a corrupted data, P23 sends the original data with the encryption key.

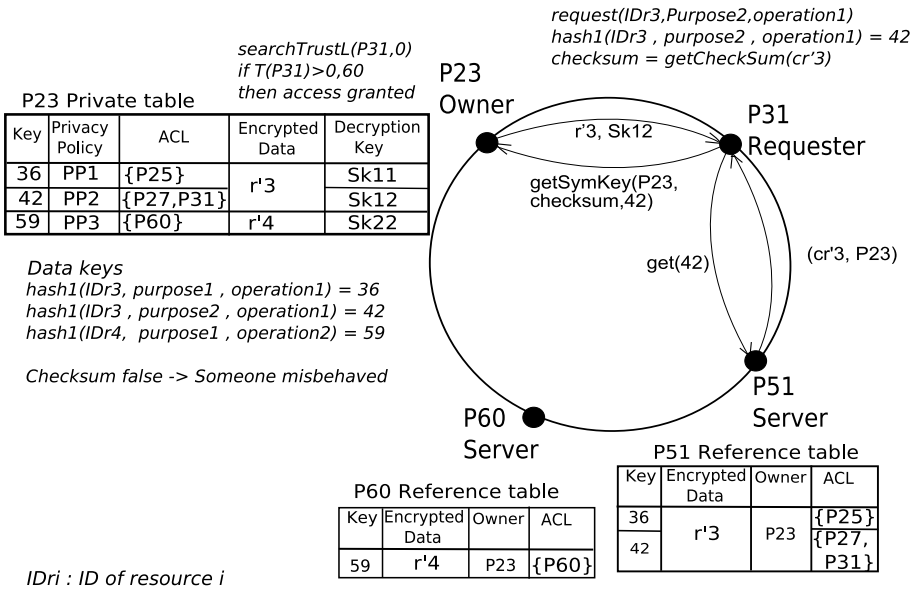


Figure 5.13: Data requesting example when data are published

5.4.3 Trust level searching

Figure 5.14 illustrates the with-friends algorithm with a maximum depth equal to 2, and where the trust level of the requesting peer (P31) is not in the trust table of the owner (P23). In order to find this level, P23 contacts its friends, in this case P51 ($searchTrustL(P31, 1)$).

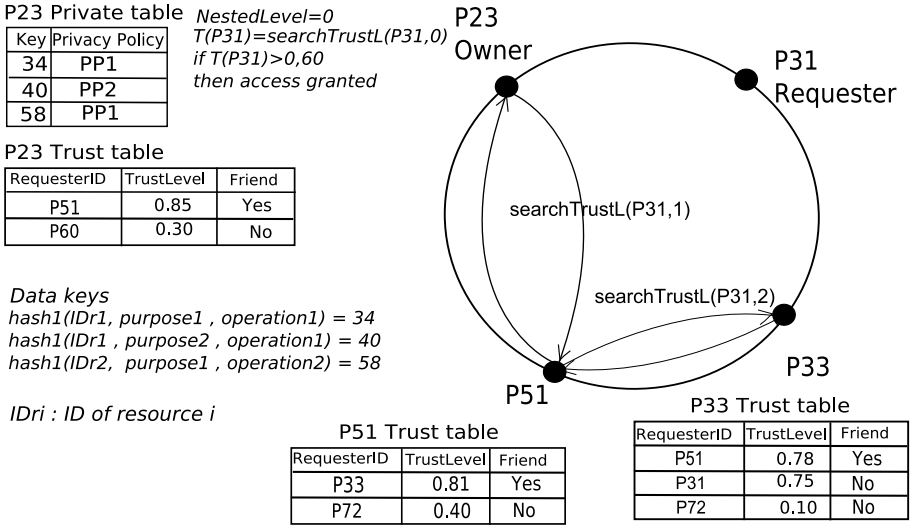


Figure 5.14: Trust level searching example

P51 does not have the trust level of P31 in its trust table so it contacts, in turn, its friends (P33). P33 finds the trust level and sends it to P51. P51 updates its trust table and sends the trust level of P31 to P23. Then P23 can grant the data access.

If a peer contacts several friends, it computes an average of the received trust levels weighted with the corresponding friend trust level. For example, in Figure 5.14, consider that P51 has a trust level equal to 0.85 and P33 has 0.80 as a trust level and the trust table of P51 has an entry for the trust level of P31 equal to 0.50. Thus, the resulting trust level of P31 computed by P23 is 0.51 (i.e., $(0.85 * 0.50 + 0.80 * 0.75)/2$).

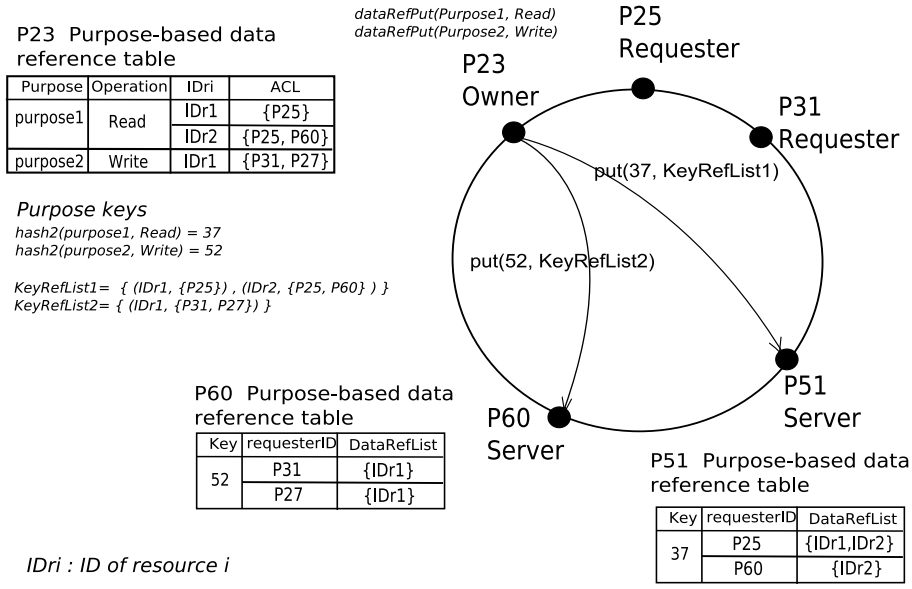


Figure 5.15: Purpose-based reference publishing example

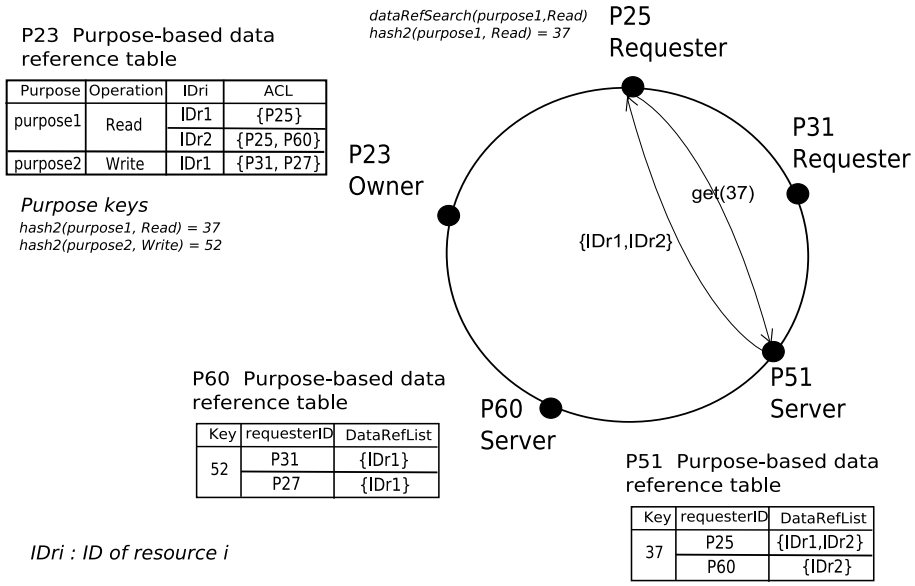


Figure 5.16: Purpose-based reference searching example

5.4.4 Purpose-based reference searching

Figure 5.15 illustrates the purpose-based reference publishing algorithm. *KeyRefList1* (resp. *KeyRefList2*) is a list of couples created by P23 from data identifiers (IDri) allowed for *purpose1* (resp. *purpose2*) and *Read* (resp. *Write*), and the corresponding ACL. *hash*2 is used to produce keys 37 and 52 from *(purpose1, Read)* and *(purpose2, Write)*. P23 publishes *KeyRefLists* by using the put function. P51 is the server peer responsible for key 37 and P60 for 52. P51 (resp. P60) reorganizes *KeyRefList1* (resp. *KeyRefList2*) in order to have for each requester identifier, its list of allowed data identifiers.

Figure 5.16 illustrates the purpose-based reference searching algorithm. P25 requests its data reference list allowed for *purpose1* and *Read* which corresponds to key 37 (*get*(37)). The peer which identifier is equal to or follows 37 is P51. P51 returns the data reference list *{IDr1, IDr2}* corresponding to key 37 and requester P25.

5.5 Conclusion

This chapter has shown PriServ a privacy service for structured P2P systems which implements PriMod. PriServ guarantees limited collection of private data since owners control the access to their private data.

First, we presented PriServ design choices. PriServ is implemented on top of DHTs and is independent of the type of DHT used. We can imagine PriServ running on whatever DHT as long as they specify methods allowing peers to put and get data from the system.

Another design choice is the integration of access purposes and operations in the creation of hash keys. This way, publishing and requesting in PriServ are always done for specific purposes and operations.

Then, we presented the PriServ architecture. It is based on several managers, each responsible of a particular task (e.g., trust, cryptography, privacy policies, logs, etc.) and an orchestrator that manages calls for these manager. Thanks to this architecture, owners are able to publish their data and to protect their data privacy from malicious peers[4].

We presented two algorithms to publish data and data references, an algorithm to request data, three algorithms to search for trust levels, and an algorithm for purpose-based sharing.

A prototype of PriServ has been developed in Java. The next chapter presents a cost analysis of PriServ. We also test PriServ by using simulations and show a demonstration of a privacy preserving application which uses PriServ.

[4]Owners can detect misbehaviors, however for the moment, they are not able to detect who misbehaved. Future work includes the design of an auditing service which allows to discover the identity of malicious peers.

Chapter 6

Validation

This chapter presents an experimental validation of PriServ through a cost simulation and a demonstration of a medical application.

6.1 Introduction

Chapter 5 have shown the architecture and functionalities of PriServ. This chapter validates PriServ. First, we show a costs analysis of PriServ main algorithms, in particular publishing, requesting and trust level searching costs. Then, we simulate PriServ functionalities in order to validate the algorithms costs. Simulations adhere to the cost analysis and show that PriServ introduces small overheads. Afterward, the simulation results motivated us to implement a prototype of PriServ and to use this prototype to built a privacy-preserving application for medical online communities.

This chapter is organized as follows. Section 6.2 analyzes PriServ main costs. Section 6.3 presents the simulation of PriServ functionalities. Section 6.4 shows the implementation of the PriServ prototype. Section 6.5 introduces privacy-preserving applications. Section 6.6 concludes.

6.2 Costs analysis

In this section, we analyze the costs of the previous algorithms in terms of number of messages. We do not analyze the join/leave cost because it is the same of Chord.

6.2.1 Publishing cost

By using the DHT, $O(logN)$ messages are needed to publish each key. In PriServ, the number of keys is equal to the number of entries (i.e., *ept*) of the private data table (cf., Section 4.5.5). Additional costs induced by the cipher key generation and the data encryption are negligible vis-a-vis the network costs. Thus, the publishing cost is:

$$C_{Publish} = \sum_{i=1}^{ept} O(logN) = O(ept * logN)$$

The maximum value of *ept* is equal to the number of shared data (*nbData*) multiplied by the number of purposes (*nbPurpose*) multiplied by the number of operation (*nbOperation*), i.e., at worst, each data item is shared for all purposes and all operations:

$$CMax_{Publish} = O(nbData * nbPurpose * nbOperation * logN)$$

We can see that the number of purposes and operations affects the publishing cost. Previous studies have shown that considering ten purposes allows to cover a large number of applications [P3P, LAP]. Used with ten purposes (by data item) and three operations, PriServ incurs a small overhead. Overall, the publishing cost remains logarithmic.

6.2.2 Requesting cost

The requesting cost is the result of two costs: DHT get() cost and the retrieving cost. We disregard access control, checksum calculation and decryption costs which are negligible vis-a-vis the network costs. The DHT get() cost is in $O(logN)$ and the server returns its answer in one message. For data retrieving, a requester needs one message to contact an owner and vice versa.

To resume, the requesting cost is:

$$\begin{aligned} C_{Requesting} &= C_{DHTGet} + C_{Retrieving} \\ &= O(logN) + 1 + 2 \\ &= O(logN) + 3 \\ &= O(logN) \end{aligned}$$

6.2.3 Trust level searching cost

The trust level searching cost (C_{STL}) is different in function of the trust searching algorithm. In the following, we show the trust level searching cost for the three algorithms seen in Section 5.3.2.2.

6.2.3.1 With-friends algorithm

In this case, the owner sends a message to each of its friends which in turn do the same in a nested search. This cost depends on the number of friends (NF) and the maximum depth of the nested search (MaxDepth).

$$C_{STL_{WF}} = \sum_{i=1}^{MaxDepth} NF^{\ i} = O(NF^{MaxDepth})$$

6.2.3.2 Without-friends algorithm

In this case, the owner sends a message to each of the peers in its finger table which in turn do the same in a nested search. This cost depends on the number of fingers which is log N and the maximum depth of the nested search (MaxDepth).

$$C_{STL_{WOF}} = \sum_{i=1}^{MaxDepth} (logN)^{\ i} = O((logN)^{MaxDepth})$$

6.2.3.3 With-or-without-friends algorithm

In this case, if the owner has friends, it sends a message to each of its friends, if not it sends a message to each of the peers in its finger table. A peer contacted by an owner does the same in a nested search. The trust level searching cost depends on the number of friends (NF), the number of fingers which is log N and the maximum depth of the nested search (MaxDepth).

$$C_{STL_{WWF}} = O((max(logN, NF))^{MaxDepth})$$

The trust level searching cost C_{STL} can be one of the three costs $C_{STL_{WF}}$, $C_{STL_{WOF}}$ or $C_{STL_{WWF}}$. Note that if NF > log N, $C_{STL_{WWF}}$ is equal to $C_{STL_{WF}}$, else it is equal to $C_{STL_{WOF}}$. In all cases $C_{STL_{WWF}}$ can be used for C_{STL}:

$$C_{STL} = O((max(logN, NF))^{MaxDepth})$$

To summarize,

$$\begin{aligned} C_{publishing} &= O(nbData * nbPurpose * nbOperation * logN) \\ C_{Requesting} &= C_{Request} + C_{Retrieve} = O(logN) \\ C_{STL} &= O((max(logN, NF))^{MaxDepth}) \end{aligned}$$

Compared to the traditional DHT functions, $C_{request}$ and $C_{retrieve}$ costs do not introduce communication overhead in term of number of messages. However, C_{STL} increases the data requesting cost due to the nested search. Next section shows how this cost can be reduced and stabilized.

Simulation parameters		
Variable	**Description**	**default**
n	Number of bits in the key/peer	11
N	Number of peers	2^{11}
FC	Number of friends	2
MaxDepth	Maximum depth of trust searching	11

Table 6.1: Table of parameters

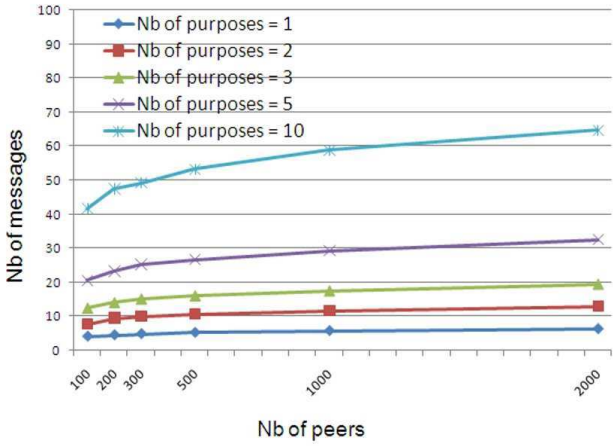


Figure 6.1: Publishing cost

6.3 Simulation

This section evaluates the performance of PriServ by simulation. We focus on data publishing (cf., Section 6.3.1), requesting (cf., Section 6.3.2) and the trust level searching (cf., Section 6.3.3) costs.

For the simulation, we use SimJava [HM98]. We simulate the Chord protocol with some modifications in the *put*() and *get*() functions. The parameters of the simulation are shown in Table 6.1. In our tests, we consider N peers with a number of data keys equal to the number of data multiplied by the number of purposes and operations. Data and peer keys are selected randomly between 0 and 2^n. In our simulation, we set n to 11 which corresponds to 2^{11} peers. This number of users is largely sufficient for collaborative applications like medical research (e.g., Doc2Doc can contain up to 3000 users per topic).

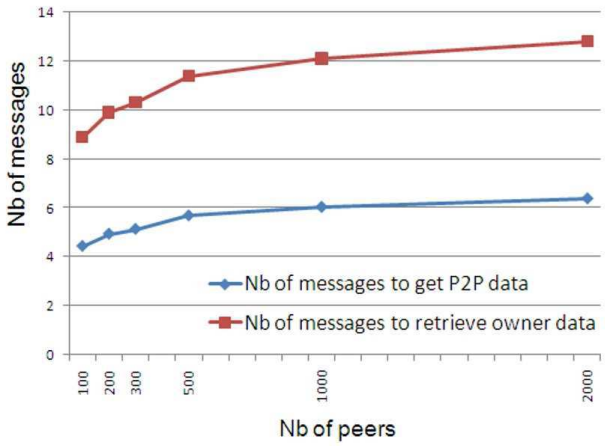


Figure 6.2: Requesting cost

6.3.1 Publishing

We measure the number of messages for publishing one data item (nbData is equal to 1) versus the number of peers. Figure 6.1 illustrates 5 measurements where the number of purposes (nbPurpose) goes from 1 to 10. Note that with only 1 purpose, the publishing cost is that of a system without the real notion of purpose. We can observe that the publishing cost is logarithmic and increases with the number of purposes. This can be explained by the fact that with the notion of purpose, we have to publish more keys. Thus, this small overhead is normal. We recall that having 10 purposes for each data item is an extreme case.

6.3.2 Requesting

We measure the number of messages for requesting one data item[1] versus the number of peers. Figure 6.2 shows two costs. The first cost is the number of messages to get the provider identifier. The second cost is the sum of the first cost and the number of messages to get the private data. We observe that the requesting costs are logarithmic as predicted by our cost model (see Section 6.2.2).

6.3.3 Trust level searching cost

We measure the trust level searching cost (in number of messages) versus the number of peers for the three algorithms seen in Section 5.3.2.2.

[1]Requesting one data item means searching for a datum for one purpose and one condition. If a requester needs to search for a data for two different purposes, she must request the same data twice. However, if she already has the data content and needs to access it for a different purpose, it is possible to extend PriServ to propose access authorization requesting without transmitting the data content.

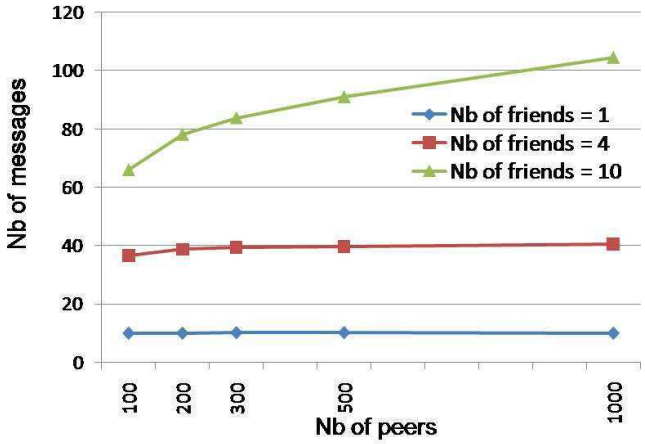


Figure 6.3: Trust level searching cost: with-friends algorithm

6.3.3.1 With-friends algorithm

We consider that peers have the same number of friends (NF) and the maximum depth ($MaxDepth$) is set to 11. This corresponds to an exhaustive search. By increasing $MaxDepth$, the probability of finding a trust level increases significantly. With $MaxDepth$ equal to 11, if NF is equal to 2, a peer could contact 2^{11} peers (i.e., the total number of peers). Thus, the probability to find the trust level becomes 1.

Figure 6.3 illustrates 3 measures where we consider 1, 4 and 10 friends. Those measures are slightly different of the cost model where the cost is $O(NF^{MaxDepth})$ thanks to our tree-based optimization: we consider the trust level searching as a search tree in which the owner is the root, the contacted peers are the nodes, each contact is an edge and the depth is equal to $MaxDepth$. In the simulation, peers who already are in the search tree will not be recontacted. The probability to contact twice a peer in a system of 100 peers is higher than in a system of 1000 peers. That is why in Figure 6.3, the trust level searching cost increases with the number of peers. We observe that for a small number of friends the trust level searching costs depends only on the number of friends as predicted by our cost model.

6.3.3.2 Without-friends algorithm

Figure 6.4 illustrates 4 measures where MaxDepth varies between 1, 2, 3 and 11. We recall that the number of contacted peers is $log(N)$. Thus, the trust level searching costs is logarithmic for small values of depth. This cost increases with the maximum depth of searching as predicted by our cost model.

6.3.3.3 With-or-without-friends algorithm

We consider that the probability that a peer has friends in our simulation is 0.9. Figure 6.5 illustrates 3 measures where the maximum depth of searching varies between 5, 8 and

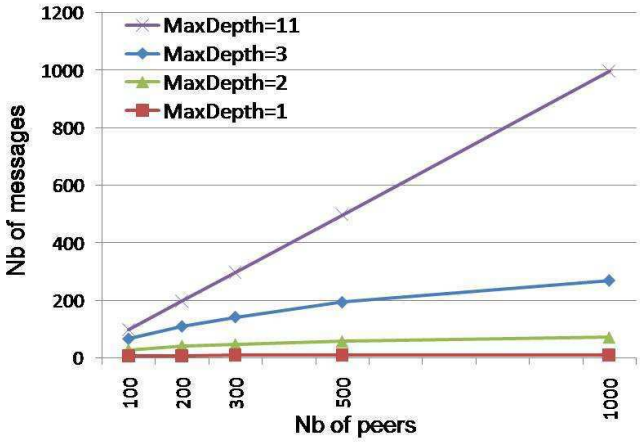


Figure 6.4: Trust level searching cost: without-friends algorithm

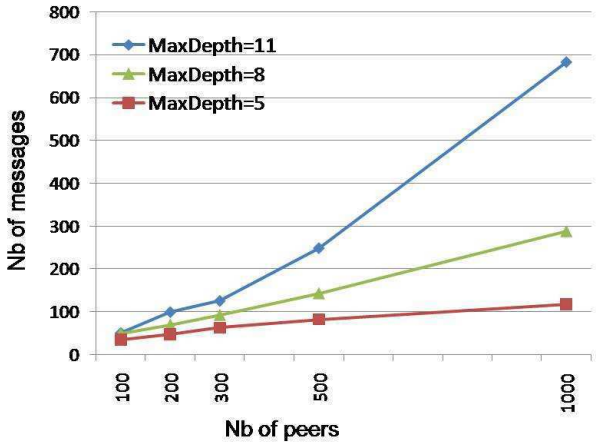


Figure 6.5: Trust level searching cost: with-or-without-friends algorithm

11. We observe that the trust level searching cost is rather logarithmic for small values of depth. This cost increases with the maximum depth as predicted by our cost model. We do not measure in this case the trust level searching cost versus the number of friends which will be the same as in the with-friends case.

6.3.3.4 Comparison

Figure 6.6 compares the three algorithms seen above. We consider a number of friends equal to 2 and a maximum depth equal to 11. As predicted before, the with-friends case introduces the smallest cost while the without-friends case introduces the highest. However, intuitively, the probability to find the trust level is higher in the without-friends algorithm than in the with-friends algorithm. This is due to the fact that the number of contacted peers is higher in the without-friends algorithm, which increases the probability

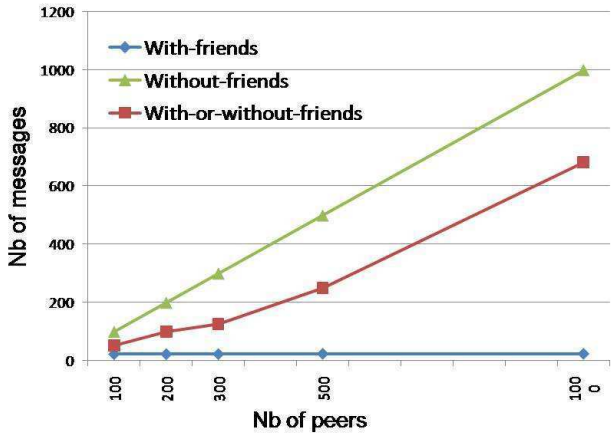


Figure 6.6: Comparison of the three algorithms of trust level searching

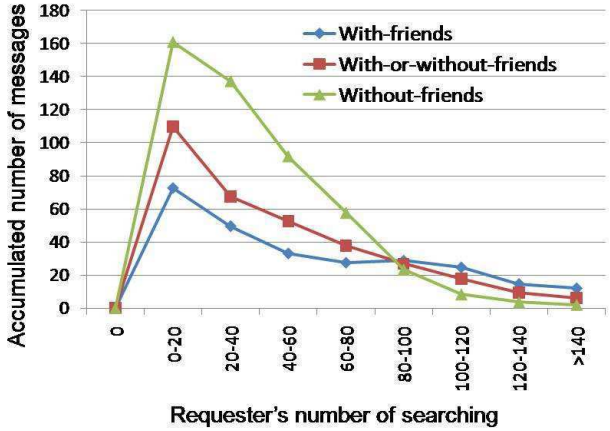


Figure 6.7: Stabilization of the cost of trust Level searching

to find the trust level. We estimate that the with-or-without-friends algorithm is the most optimized because it is a tradeoff between the probability to find the requester trust level and the trust level searching cost.

6.3.3.5 Stabilization of the cost of trust level searching

We now focus on the number of messages used to search the trust level of a requesting peer versus the number of its requests. Here we consider the three algorithms of trust (see Figure 6.7). We observe that the number of messages decreases and stabilizes after a number of searches. This is due to the fact that the more a peer requests for data, the more it gets known by the peers in the system.

When peers ask for a trust level, answers are returned in the requesting order and the trust tables are updated with the missing trust level. Thus, the trust tables evolve with

the number of searches. After a while, these tables stabilize. Thus, the number of messages for searching trust levels is reduced to a stable value. This value is not null because of the dynamicity of peers. In our simulation, we consider that the number of peers joining the system is equal to those leaving the system. Thus, there are always new peers which do not know the requester trust level.

We also observe in Figure 6.7, that the trust level searching cost in the without-friends algorithm stabilizes first. This is due to the fact that a larger number of peers are contacted in this algorithm. The with-or-without-friends algorithm comes in second place, and the with-friends algorithm comes last. As we have seen in the comparison of the three algorithms, we find again that the with-or-without-friends algorithm is the most optimized because it is a tradeoff between the time to stabilization and the trust level searching cost.

6.4 PriServ prototype

In the previous sections, we have tested PriServ functionalities by using SimJava and shown by simulation that PriServ is scalable. To test and validate its feasibility, we implemented a Java prototype for privacy-preserving data sharing applications for online communities.

6.4.1 Implementation tools

Eclipse Galileo [GAL] is used as a development framework. The language Java, SCA tools [Cha07] and Java RMI [RMI] are used to implement the prototype. We choose these tools for their simplicity and portability. In a heterogeneous environment like P2P networks, portability is required to guarantee the running of PriServ independently from its execution framework. In the following, we give an overview of SCA and JAVA RMI.

6.4.1.1 Service component architecture

The Service Component Architecture (SCA) approach provides a programming model for building applications and systems based on a Service Oriented Architecture (SOA) [Erl05]. It is based on the idea that the business function is provided as a series of services, which are assembled together to create solutions that serve a particular business need. These composite applications can contain both new services created specifically for the application and also business function from existing systems and applications, reused as part of the composition. SCA provides a model both for the composition of services and for the creation of service components, including the reuse of existing application functions within SCA compositions.

SCA divides up the steps in building a service-oriented application into two major parts:

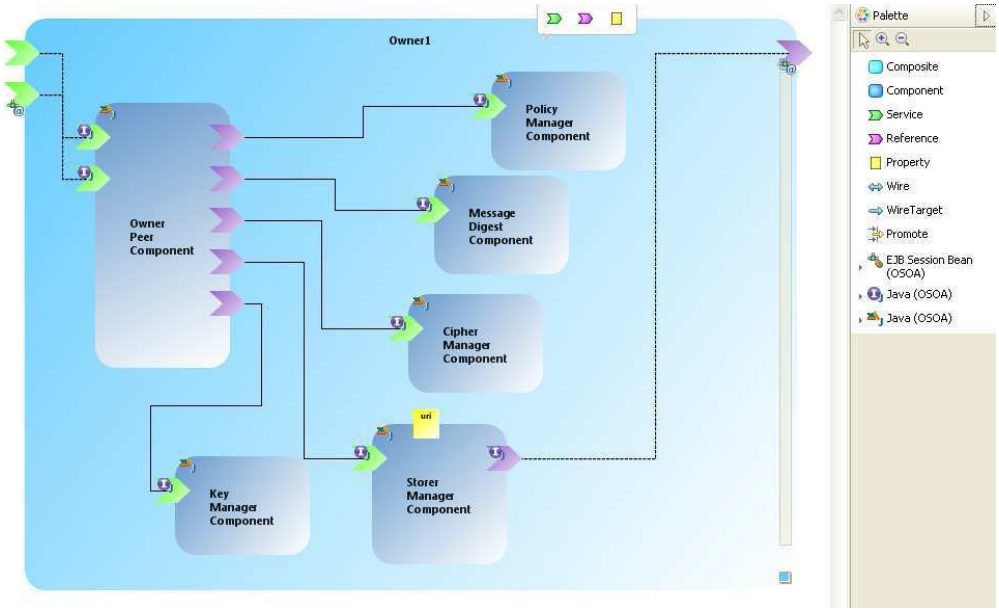


Figure 6.8: Owner peer modeled with SCA

- The implementation of service components which provide services and consume other services.
- The assembly of sets of components to build business applications, through the wiring of service references to services.

SCA supports service implementations written in several programming languages such as Java, PHP, C++, etc. A SCA plug-in for Eclipse Galileo is available and can be downloaded from [ECL].

Thanks to SCA, the implementation and the connection of PriServ components were efficient and easy. Figure 6.8 shows the owner peer implementation in SCA.

6.4.1.2 Remote method invocation

The Java Remote Method Invocation (RMI) system allows an object running in one Java virtual machine (JVM) to invoke methods on an object running in another JVM. RMI provides for remote communication between programs written in the Java programming language.

Objects that can be invoked across JVMs are called *remote objects*. An object becomes remote by implementing a *remote interface*, which has the following characteristics:

- A remote interface extends the interface java.rmi.Remote.
- Each method of the interface declares java.rmi.RemoteException in its throws clause, in addition to any application-specific exceptions.

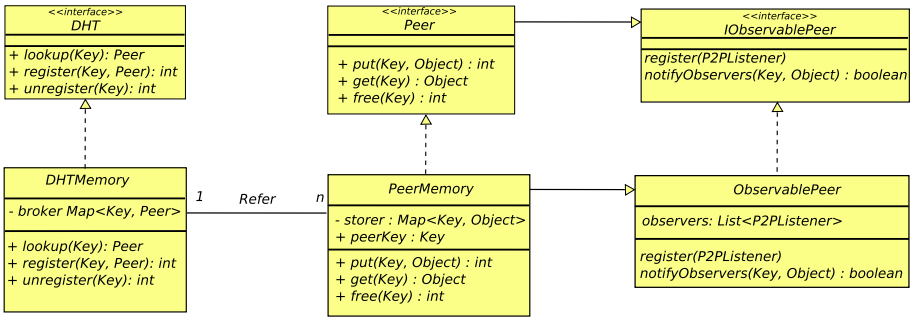


Figure 6.9: Class diagram of the DHT

RMI treats a remote object differently from a non-remote object when the object is passed from one JVM to another JVM. Rather than making a copy of the implementation object in the receiving JVM, RMI passes a remote *stub* for a remote object. The stub acts as the local representative, or proxy, for the remote object and basically is, to the client, the remote reference. The client invokes a method on the local stub, which is responsible for carrying out the method invocation on the remote object.

In the PriServ prototype, Java RMI is used to link the DHT peers and to link owners to requesters while retrieving data. Owner identifiers are remote stubs, sent to requesters, on which they can invoke the retrieve function in order to retrieve the data.

Java RMI is included with Java SE and is available as a separate download for Java ME.

6.4.2 Prototype implementation

We performed the PriServ project in two phases: the implementation phase and the test phase.

6.4.2.1 Implementation phase

We devided the implementation phase into two subproject: the DHT implementation, and the components implementation.

DHT implementation. As mentioned before, PriServ is independent of the DHT choice. The DHT developed for the prototype (cf., Figure 6.9) was designed in order to provide peers localization, get and put functions and data storage. The *DHTMemory* class manages peer localization by storing couples (key, peer). From a key, users can find the peer responsible of a datum. The *PeerMemory* class provides get and put functions in order to store and return data for particular hash keys. Data storage is managed by the class *PeerMemory* who stores (key, object) couples.

Components implementation. Components implementation includes managers implementation and orchestrator implementation.

- Managers implementation. Each manager is responsible for simple tasks such as data storage, policy management, key creation, etc. PriServ managers[2] (i.e., Storage manager, cipher manager, policy manager, key manager, id manager, time manager, log manager and checksum manager) have been implemented as following. For each manager, we created a package containing all the classes required to insure the manager functionalities. For instance, the policy manager package contains classes for all the policy model elements (i.e., classes *Purpose*, *Operation*, *Conditions*, *ACL*, etc.) in addition to the main class *PrivacyPolicyManager* which defines methods for adding, searching and checking privacy policies. Figure 6.10 shows the policy manager package.
- Orchestrator implementation. We defined three types of orchestrators (i.e., Owner, Requester, Server). The class *Owner* orchestrates the owner's managers in order to publish data and references on the servers. The class *RemoteOwnerPeerProxy* is used to retrieve data from owners. The class *Requester* orchestrates the requester's managers in order to search for data. Finally, the implementation of the server orchestrator, who is responsible of verifying the access rights of requesters, was a challenge. In order to use any DHT without any modification, we should not modify the *PeerMemory* class. Thus, we proposed a listener (i.e., class *ObservablePeer* in Figure 6.9) which invokes the verify method each time the get function is called.

6.4.2.2 Test phase

The test phase consisted on three types of tests: the unit tests, the integration tests and the validation tests.

Unit tests. We tested the behavior of each one of the PriServ components and the DHT elements by using the JUnit framework [JUN]. JUnit is a simple framework to write repeatable tests. We use assertions in order to verify if the results returned by each method correspond with the results expected from this method.

Integration tests. We tested the interaction between PriServ components, orchestrators and the DHT. We used test scenarios such as publishing, requesting, etc. For instance, to test publishing data for particular purposes and operations, the *Owner* orchestrator creates a privacy policy by invoking the *PolicyManager* methods. Then, it encrypts data by invoking the *CipherManager* methods. Furthermore, it invokes the *distributedPut* method of the *StorageManager* which invokes the put method of the DHT *PeerMemory*. We then use the *Requester* orchestrator to search for the data and compare the result with the local data stored by the owner.

[2]Only the trust manager has not been implemented. Trust management in PriServ needs more research and is part of future work.

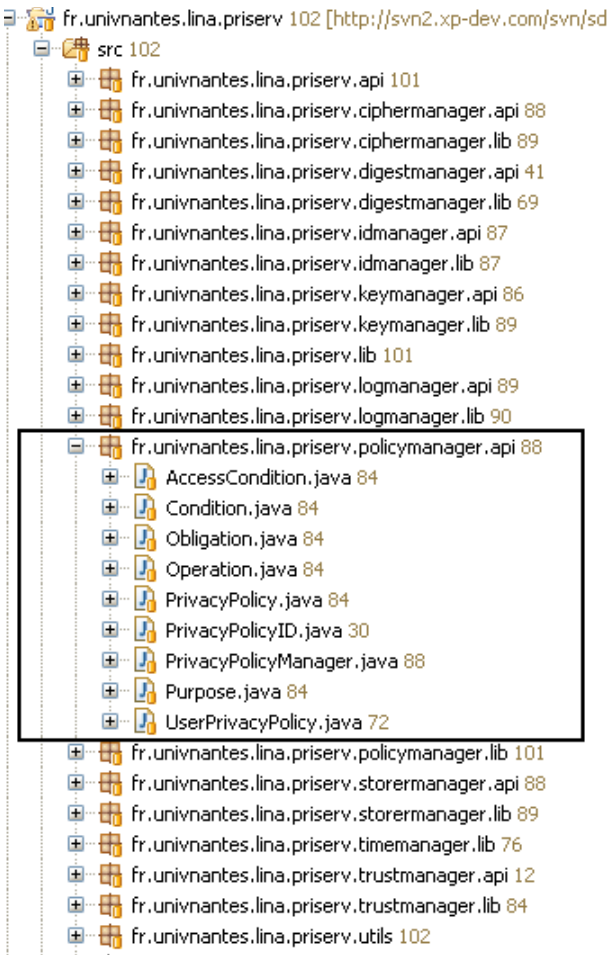


Figure 6.10: Managers implementation: Policy manager package

Validation tests. PriServ was tested and validated by PeerUnit [PEE] on Grid5000 [GRI] in [Luc10].

PeerUnit is a testing framework for large-scale distributed systems. It is useful to developers who want to test their Java applications in a distributed way. Since PriServ is intended to be used for P2P applications, PeerUnit have been used to test multiple instances of PriServ working together.

Grid5000 is a scientific instrument for the study of large scale parallel and distributed systems. It aims at providing a highly reconfigurable, controllable and monitorable experimental platform. The infrastructure of Grid5000 is geographically distributed on different sites, nine in France and one in Porto Alegre, Brazil.

The tests were done on a population of 180 peers on 42 Grid5000 nodes. Several results have validated the performance of the prototype concerning:

- Respect of owners privacy policies. PriServ limits data access if a requester does not intend to respect the owner privacy policy. The test results show the failure of the

request each time the access purpose or operation are different from the intended purpose and operation specified while publishing data.

- Limited access only for authorized requester. The test results show that 100 % of queries from an unauthorized requesters are systematically rejected by servers.

- Traceability of the data distribution. The test results show that it is possible to review, automatically, all the requesters who had access to data, and to check the list of those who have tried to access them. This allows the data owner to be sure that its data privacy requirements are respected.

6.5 Privacy-preserving applications

Privacy-Preserving Applications (PPA) are applications which manage sensitive data[3] (e.g., health care data, etc.). In general, PPAs are developed on trusted servers for closed professional communities such as medical or research communities.

The following shows how PriServ can be used in a medical PPA to publish and to request data according to doctors and patients' privacy preferences.

6.5.1 Medical PPA

Recall the collaborative medical research application presented in Section A.1. This application relies on patient characteristics (e.g., age, diet, gender, etc.). The participants are scientists, doctors, and patients. In order to control disclosure of sensitive files (e.g., medical records owned by doctors) without violating privacy, files access should respect the privacy preferences of their owners. In this medical PPA:

- Patients and doctors who own and manage private medical records can be considered as owners. Scientists who may use medical records for scientific research can be considered as requesters. Servers are peers of the storage system.

- Doctors may define the privacy preferences of patients in privacy policies and attach them to their medical records. For instance, a doctor may allow writing access on her information to scientists for adding comments on her diagnosis.

PriServ is used for these scenarios:

- After defining their privacy preferences on their medical records, doctors and patients can publish their data in the system while preserving their privacy.

[3]Many projects propose solutions for modeling PPAs. DEMOTIS [DEM] brings together computer scientists and legal scholars. DEMOTIS stands for Collaborative Analysis, Evaluation and Modelling of Health Information Technology. The project experiments new methods for the multidisciplinary design of large information systems that have to take in account legal, social and technical constraints. Its main field of application is personal health information systems.

- Scientists can search for data using procedures that respect the privacy preferences of the data owners.

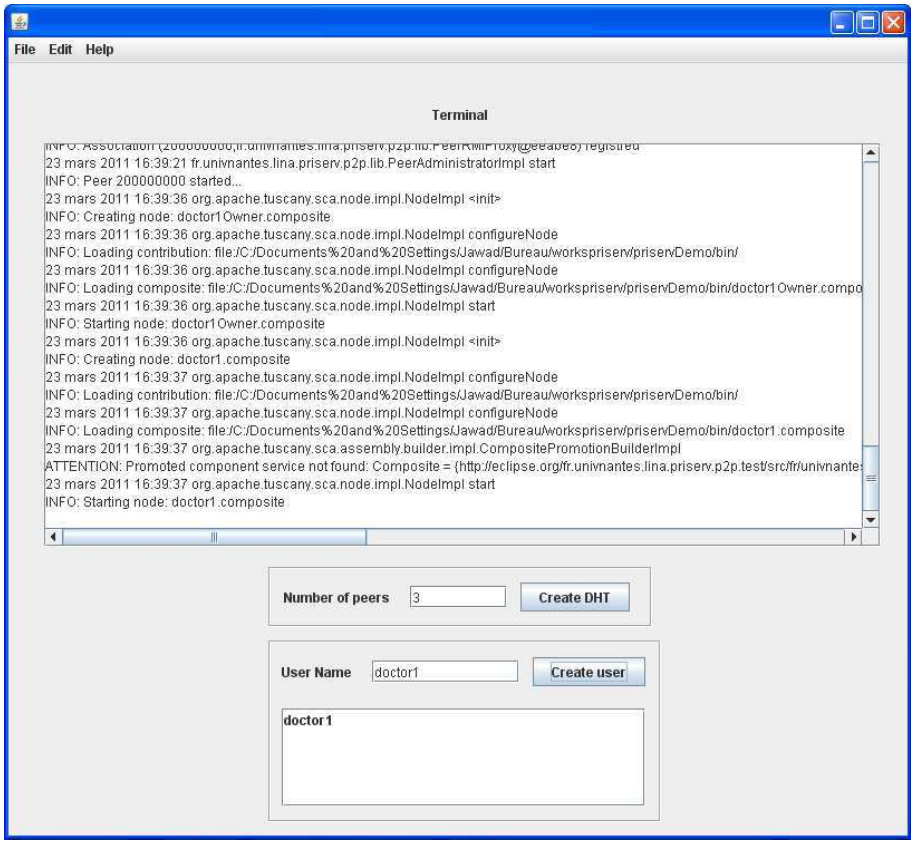


Figure 6.11: Medical PPA: DHT interface

Through a GUI, we show several scenarios that exhibit important aspects of private data management: privacy policy management, data publishing, data searching, and data reference searching. Figure 6.11 shows the interface through which the DHT is launched. Once launched, the DHT creates storer peers whose number is specified by the demonstrator. Then the demonstrator creates owners and requesters by specifying their user names and clicking on *create user*. The interface of the user newly created will appear automatically. For instance, Figure 6.12 shows the user interface of doctor1.

6.5.2 Scenarios

The key features of the prototype, are demonstrated through the following scenarios of the medical PPA:

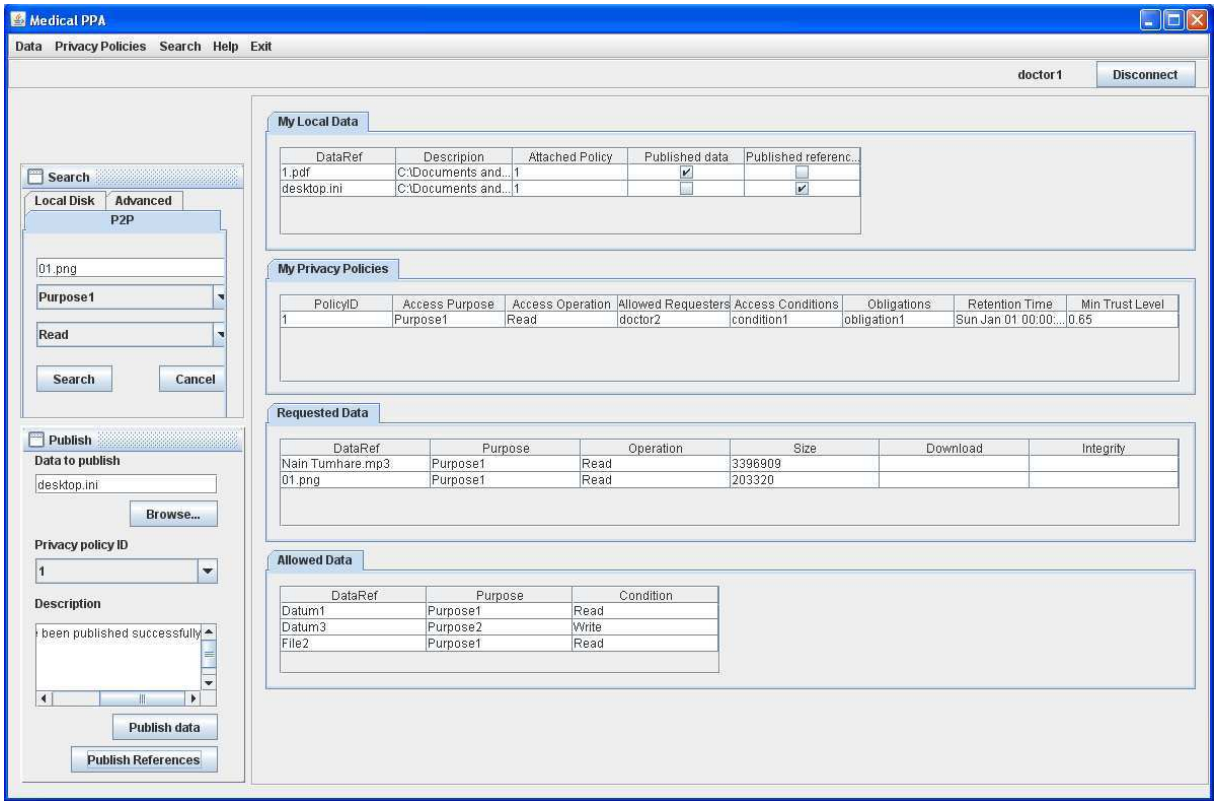


Figure 6.12: Medical PPA: doctor1 user interface

Privacy policies management. This scenario is used to show how an owner could specify her privacy preferences by defining her own privacy policies (cf., Figure 6.13). She is also able to attach different policies to a datum in order to control the access to her data. She also has an interface that shows information about her privacy policies.

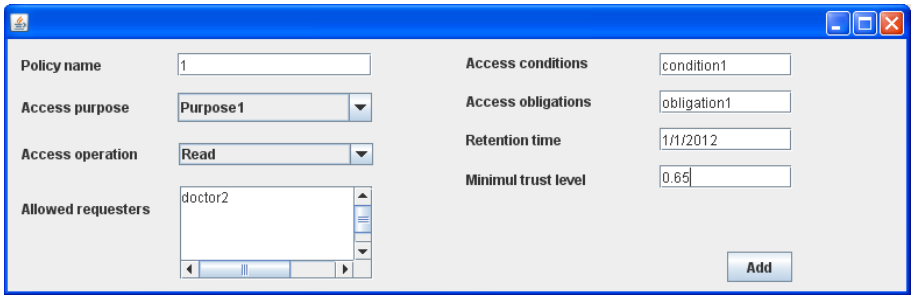
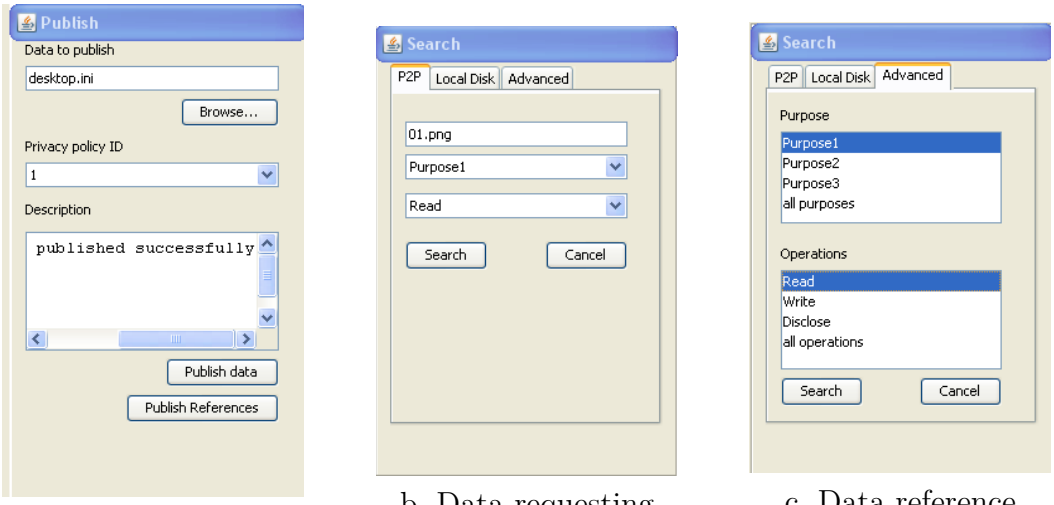


Figure 6.13: Medical PPA: Policy specification interface



a. Data publishing interface

b. Data requesting interface

c. Data reference searching interface

Figure 6.14: Medical PPA

Data publishing. This scenario shows how a user can correctly and securely publish sensitive data in the P2P system (cf., Figure 6.14.a). A user can specify for which privacy policy her data are published. She also has the choice between publishing encrypted data or data references. She can also have a log of her published data and the policies attached to them.

Data searching. This scenario shows how a user (e.g, scientists) can search for data (cf., Figure 6.14.b). A user has a choice between: (a) a local search on her own local data, (b) a P2P data requesting. We show how data access is only granted to allowed requesters. For this, we prove that a user will not be able to access data for which she does not have access rights. We also show that an authorized user will have access to valid data even if they were modified maliciously in the P2P system server.

Data reference searching. This scenario shows how a user (e.g, scientists) can search for data references for particular purpose and operation (cf., Figure 6.14.c). For this, we show that a user will be able to have a list of references for data which she can access for particular purpose and operation. Then, this user can access data content by specifying the data reference that she has gotten, the purpose and the operation in the data searching interface.

6.6 Conclusion

This chapter has shown the PriServ experimental validation. The costs analysis and PriServ simulation with SimJava revealed that publishing data in the system conserves the logarithmic cost of the traditional put function. Concerning requesting costs, trust level searching introduces a large overhead, otherwise costs are logarithmic like in the traditional get function. However, we showed that the trust level searching cost decreases with the number of request and stabilizes.

A prototype of PriServ is implemented in Java by using SCA tools and RMI. The DHT design and implementation was done in collaboration with Stéphane Drapeau from Obeo. All of PriServ components were developed by us except the log, id and time managers which have been developed by Boris Lucas during his research internship as a master student. The prototype has been tested and validated on Grid5000 by using PeerUnit. The test results motivated us to built a privacy-preserving sharing application for medical data by using the PriServ prototype.

The PriServ web site is *http://atlas.lina.univ-nantes.fr/gdd/appa/priserv* and the prototype code can be found at *http://sourceforge.net/projects/priserv/*.

Chapter 7

Conclusion

Within just a few years, P2P systems gained a huge popularity and drew the attention of data-centered applications due to their efficient characteristics (e.g., scalability, distribution, autonomy, etc.). However, their limited guarantees concerning data privacy still discourage some applications aiming at sharing sensitive data. P2P systems could be considered as a source of privacy risks because, in principle, each peer in the system has access to shared data and they can use them at their will for different purposes.

In this context, the main goal of this thesis was to contribute to the development of a novel and efficient privacy service for P2P systems. Other goals were to promote the use of P2P systems among data-centered applications by enforcing privacy protection, and to compel P2P users to respect privacy laws and legislations.

This thesis presented an integrated solution for protecting data privacy in P2P systems according to data owners' privacy preferences. After analyzing existing works on privacy techniques and comparing current P2P systems with respect to privacy protection, a privacy model and a privacy service for P2P systems were proposed and implemented. Section 7.1 gives a summary of the main contributions of this work. Then, Section 7.2 introduces some future works.

7.1 Summary of contributions

This section resumes the main contributions of this work: a state-of-the-art on data privacy (cf., Section 7.1.1), PriMod, a privacy model for P2P systems (cf., Section 7.1.2), and

PriServ, a privacy service for DHT-based P2P systems (cf., Section 7.1.3)

7.1.1 State-of-the-art

The first contribution aimed at a comprehensive survey of data privacy and P2P systems, in order to motivate the next contributions. The state of the art was conducted into two phases.

First, a survey of privacy protection techniques helped to identify several ways for protecting privacy: (a) access control techniques which allow to limit unauthorized access to private data, (b) Hippocratic databases which integrate the notion of *purposes* to privacy protection, (c) anonymity techniques which protect privacy by making users indistinguishable of others and hardly identified, (d) trust techniques which predict the behavior of unknown users and thus provide protection from malicious acts, and (e) cryptography techniques which protect data privacy by making them unreadable by unauthorized users.

Second, a survey of privacy services for P2P systems compared existing solutions based on the privacy techniques used to protect data privacy (i.e., access control, anonymity, trust and cryptography) and the privacy properties guaranteed (i.e., protection against unauthorized reads, corruption and deletion, limited disclosure, anonymity guarantees, denial of linkability and content deniability).

The first observation from the state-of-the-art was that combining privacy techniques increases the level of privacy. However, some techniques can not easily be combined such as anonymity techniques which hide users identities, and access control techniques which need users identities to limit unauthorized access. The second observation was that existing P2P systems do not use completely all the privacy techniques, and neither of them guarantees all the privacy properties we identified at once. The third observation was that, while the notion of purpose has a relevant importance, as accentuated by the OECD guidelines, Hippocratic principles and P3P specifications, it has not been used yet in P2P systems. Thus a new privacy solution for P2P systems that uses purpose-based access control was needed.

7.1.2 PriMod, a privacy model for P2P systems

Motivated by the necessity of purpose specification, the second contribution of this work aimed at proposing a new privacy model for P2P systems. PriMod was proposed to benefit from P2P efficient functionalities for data publishing and sharing while protecting data privacy. It proposes a privacy policy model which allows owners to define their privacy preferences in privacy policies, a trust model to predict peers behavior, and a data model in which sensitive data are associated with privacy policies. PriMod also offers operations for publishing data content, publishing references, requesting, and purpose-based searching. Requests are always made for particular purposes and operations which commits requesters to respect their intended use of data.

PriMod protects sensitive shared data against malicious reads, corruptions and deletions by using encryption and data checksums. Data disclosure is limited by using access control techniques. PriMod does not guarantee anonymity since it is not designed to protect users behavior but focuses on data privacy protection. Unlike all the models found in the literature, PriMod includes the notion of purpose by implying Hippocratic principles to the data model.

7.1.3 PriServ, a privacy service for DHT-based P2P systems

The third contribution aimed at building a privacy service on top of DHT-based P2P systems. Based on PriMod, PriServ prevents privacy violation by limiting malicious data access. It combines purpose and operation based access control, trust techniques, cryptography techniques and digital checksums and proposes privacy-preserving publishing and requesting functions. PriServ can be run on whatever DHT as long as they propose methods allowing peers to put and get data from the system. Publishing and requesting in PriServ are always done for specific purposes and operations. This commits requesters to use data only for the specified purposes and to perform only specified operations. Legally, this commitment may be used against malicious requesters if it is turned out that obtained data have been used differently.

PriServ introduces an acceptable overhead on DHT Systems. The costs analysis and simulation revealed that publishing and requesting costs are logarithmic like in the traditional DHTs. However, trust level searching in PriServ introduces a linear overhead, thus more research in this area is needed to optimize this cost.

Initially, we have simulated PriServ functionalities in order to validate the algorithms costs. Afterward, we implemented a prototype of PriServ. This prototype was tested and validated by PeerUnit on Grid5000 and was used to built a privacy-preserving application for medical environments.

7.2 Future work

Future work focuses on providing PriServ with more advanced features and extending it to other contexts. This section presents a non exhaustive list of works which will be the continuity of this thesis.

7.2.1 Auditing

In order to protect data privacy, it is necessary, not only to allow data owners to prevent misbehavior and to specify the way in which their sensitive data should be accessed but also to provide tools to audit privacy, that is, to verify the compliance of the data use with the specified privacy preferences. PriServ architecture contains a log manager which keeps a log for each peer in the system. Each action concerning data sharing is written down in the corresponding log. Future work focuses on designing and building an audit service for PriServ which uses peers logs to form audit queries. The idea is to add an audit

manager to the PriServ architecture in charge of collecting information to write into log records and, later, to be able to recover them.

7.2.2 Requester groups and trust management

In PriServ, when encrypted data content is published, owners must be contacted by requesters in order to retrieve decryption symmetric keys. An improvement to PriServ is to avoid bottlenecks on owners in order to have more scalability. For this, asymmetric keys can be used for encrypting data content which avoids symmetric keys exchange. In order to avoid having a large number of asymmetric keys generated, requesters could be organized in groups and each group will be assigned a pair of public/private key.

Trust management could also be done within a group. A requester needs to have a minimal trust level to make part of a group. If a requester is in a group, it could access the sensitive data allowed for this group. When a requester misbehaves, its trust level will be decremented. When the requester trust level becomes lower than the minimal trust level of a group, the group members will vote to eject the requester out of their group.

This proposal leads to other research issues such as group definition and maintenance, trust levels update, group consensus, etc., which require more research studies.

7.2.3 P2PWeb

The P2PWeb project goal is to design and develop a system for multimedia content distribution in a web environment by using P2P techniques. This project aims to study P2P technologies in order to built a system that allows users to retrieve, via their Internet Explorer, multimedia content not only from the content provider server but also from users who already have that content. In this context, privacy control and satisfaction management are essential and form a major axis of the project. The contributions of this thesis (i.e., state-of-the-art, PriMod, PriServ and its Prototype) give a solid base to propose solutions for privacy protection of multimedia content in P2P environments.

Bibliography

[ABF+04] Rakesh Agrawal, Roberto Bayardo, Christos Faloutsos, Jerry Kiernan, Ralf Rantzau, and Ramakrishnan Srikant. Auditing Compliance with a Hippocratic Database. In *Very Large Databases (VLDB)*, 2004.

[ABG+05] Rakesh Agrawal, Paul Bird, Tyrone Grandison, Jerry Kiernan, Scott Logan, and Walid Rjaibi. Extending Relational Database Systems to Automatically Enforce Privacy Policies. In *IEEE Conference on Data Engineering (ICDE)*, 2005.

[ACK+02] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: an Experiment in Public-Resource Computing. *Communications of the ACM*, 45(11), 2002.

[ACT] Microsoft Active Directory. http://www.microsoft.com/windowsserver2008/.

[AES01] Federal Information Processing Standards Publication, 2001.

[AKSX02] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Hippocratic Databases. In *Very Large Databases (VLDB)*, 2002.

[Ale10] Keith B. Alexander. National Information Assurance (IA) Glossary - CNSS-4009. 2010.

[AMPV06] Reza Akbarinia, Vidal Martins, Esther Pacitti, and Patrick Valduriez. Design and Implementation of APPA. *Global Data Management (Eds. R. Baldoni, G. Cortese, F. Davide), IOS Press*, 2006.

[ANT] The Ants P2P Web Site. http://antsp2p.sourceforge.net/.

[APV06] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. Reducing Network Traffic in Unstructured P2P Systems Using Top-k Queries. *Distributed and Parallel Databases*, 19(2-3), 2006.

[ATS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4), 2004.

[BAH+06] Rolando Blanco, Nabeel Ahmed, David Hadaller, L. G. Alex Sung, Herman Li, and Mohamed Ali Soliman. A Survey of Data Management in Peer-to-Peer Systems. Technical report, School of Computer Science, University of Waterloo, 2006.

[BBL05] Ji-Won Byun, Elisa Bertino, and Ninghui Li. Purpose Based Access Control of Complex Data for Privacy Protection. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, 2005.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized Trust Management. In *IEEE Symposium on Security and Privacy*, 1996.

[BIT] The BitTorrent web site. http://www.bittorrent.com/.

[BL73] D Elliott Bell and Leonard J LaPadula. Secure Computer Systems: Mathematical Foundations and Model. *The MITRE Corporation Bedford MA*, 1(M74-244), 1973.

[BL08] Jiwon Byun and Ninghui Li. Purpose based Access Control for Privacy Protection in Relational Database Systems. *The Very Large Database (VLDB) Journal*, 17(4), 2008.

[BLO] The Block Cipher Lounge. http://www2.mat.dtu.dk/people/Lars.R.Knudsen/bc.html.

[Blo64] Edward Bloustein. Privacy as an Aspect of Human Dignity, 1964.

[BP09] Ronald H. Brown and Arati Prabhakar. *Digital Signature Standard (DSS)*. Federal Information Processing Standards, 2009.

[CAMN03] Francisco Matias Cuenca-Acuna, Richard P. Martin, and Thu D. Nguyen. Autonomous Replication for High Availability in Unstructured P2P Systems. In *IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2003.

[CDG+02] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *Operating Systems Design and Implementation (OSDI)*, 2002.

[CDM+09] David R. Choffnes, Jordi Duch, Dean Malmgren, Roger Guiermà, Fabiàn E. Bustamante, and Luis Amaral. SwarmScreen: Privacy Through Plausible Deniability in P2P Systems. Technical report, Northwestern EECS University, 2009.

[CFB04] Vicent Cholvi, Pascal Felber, and Ernst Biersack. Efficient Search in Unstructured Peer-to-Peer Networks. *European Transactions on Telecommunications*, 15(6), 2004.

[Cha81] David L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2), 1981.

[Cha07] David Chappell. Introducing SCA, 2007.

[CJM96] Don Coppersmith, Donald Byron Johnson, and Stephen M. Matyas. A Proposed Mode for Triple-DES Encryption. *IBM Journal of Research and Development*, 40(2), 1996.

[CLM+02] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. The Platform for Privacy Preferences (P3P1.0) Specification, 2002.

[CMH+02] Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. Protecting Free Expression Online with Freenet. *IEEE Internet Computing*, 6(1), 2002.

[CSR] Computer Security Resource Center. National Institute of Standards and Technology. http://csrc.nist.gov/.

[DAT10] *Dictionary of Accounting Terms*. Barron's Educational Series, 2010.

[Dav76] Ruth M. Davis. The Data Encryption Standard in Perspective. *Communications Magazine, IEEE*, 16(6), 1976.

[DDCdVPS03] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. Managing and Sharing Servents Reputations in P2P Systems. *IEEE Transactions on Data and Knowledge Engineering*, 15(4), 2003.

[DdVP+02] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A Reputation-based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In *ACM Conference on Computer and Communications Security (CCS)*, 2002.

[DEM] The DEMOTIS project for Collaborative Analysis, Evaluation and Modelling of Health Information Technology. http://www.demotis.org/.

[DFM00] Roger Dingledine, Michael J. Freedman, and David Molnar. The Free Haven Project: Distributed Anonymous Storage Service. In *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[DGMY03] Neil Daswani, Hector Garcia-Molina, and Beverly Yang. Open Problems in Data-Sharing Peer-to-Peer Systems. In *International Conference on Database Theory (ICDT)*, 2003.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6), 1976.

[DOC] Doc2Doc, Doctors Community, Forums and Doctors Networking. http://doc2doc.bmj.com/.

[DZD+03] Frank Dabek, Ben Y. Zhao, Peter Druschel, John Kubiatowicz, and Ion Stoica. Towards a Common API for Structured Peer-to-Peer Overlays. In *International Peer To Peer Systems Workshop (IPTPS)*, 2003.

[ECL] The Eclipse SCA plugin. http://www.eclipse.org/stp/sca/index.php.

[Erl05] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, 2005.

[FED] Federal Standard 1037C. Telecommunications: Glossary of Telecommunication Terms. http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm.

[FRE] The FreeBSD Web Site, http://www.freebsd.org.

[Fur05] Pedro Furtado. Schemas and Queries over P2P. In *Database and Expert Systems Applications (DEXA)*, 2005.

[FWCY10] Benjamin C. M. FUNG, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-Preserving Data Publishing: A Survey of Recent Developments. *ACM Computing Surveys*, 42(4), 2010.

[GAL] Eclipse Galileo. http://www.eclipse.org/galileo/.

[GK03] Nathaniel Good and Aaron Krekelberg. Usability and Privacy: a Study of Kazaa P2P File-Sharing. In *Conference on Human Factors in Computing Systems (CHI)*, 2003.

[GNU] The Gnutella Web Site. http://rfc-gnutella.sourceforge.net/.

[GRE84] *Liddell and Scott's Greek-English Lexicon*. Oxford University Press, 1984.

[GRI] The Grid'5000 Web Site. http://www.grid5000.fr/.

[GRO] The SharePoint Workspace 2010. http://office.microsoft.com/en-us/sharepoint-workspace/.

[GS00] Tyrone Grandison and Morris Sloman. A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.

[HHL+03] Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, and Ion Stoica. Querying the Internet with PIER. In *Very Large Databases (VLDB)*, 2003.

[HM98] Fred Howell and Ross McNab. Simjava: a Discrete Event Simulation Library for Java. In *Society for Computer Simulation (SCS)*, 1998.

[Hof77] Lance J. Hoffman. *Modern methods for computer security and privacy.* Prentice-Hall, 1977.

[HR02] Steven Hand and Timothy Roscoe. Mnemosyne: Peer-to-Peer Steganographic Storage. In *International Peer To Peer Systems Workshop (IPTPS)*, 2002.

[IBM] IBM Hippocratic Database (HDB) Technology Projects. http://www.almaden.ibm.com/cs/projects/iis/hdb/hdb_projects.shtml.

[IPKA09] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson. Privacy-Preserving P2P Data Sharing with OneSwarm. Technical report, University of Washington, 2009.

[JAB] The Jabber Web Site. http://www.jabber.com/.

[JUN] The JUnit Framework. http://www.junit.org/.

[Kah74] David Kahn. *The Codebreakers - The Story of Secret Writing* . Macmillan Publishers, 1974.

[Kar85] Paul A. Karger. Authentication and Discretionary Access Control in Computer Networks. *Computer Networks and ISDN Systems*, 10(1), 1985.

[KAZ] The Kazaa Web Site. http://www.kazaa.com/.

[KBC+00] John Kubiatowicz, David Bindel, Yan Chen, Steven E. Czerwinski, Patrick R. Eaton, Dennis Geels, Ramakrishna Gummadi, Sean C. Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Y. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[KCFP01] Lalana Kagal, Scott Cost, Timothy Finin, and Yun Peng. A Framework for Distributed Trust Management. In *Autonomy, Delegation and Control (IJCAI)*, 2001.

[KEFP02] James H. Kaufman, Stefan Edlund, Daniel A. Ford, and Calvin Powers. The Social Contract Core. In *ACM World Wide Web Conference (WWW)*, 2002.

[KPR01] Jon Kleinberg, Christos H. Papadimitriou, and Prabhakar Raghavan. On the Value of Private Information. In *Theoretical Aspects of Rationality and Knowledge (TARK)*, 2001.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *ACM World Wide Web Conference (WWW)*, 2003.

[KSW02] Günter Karjoth, Matthias Schunter, and Michael Waidner. Platform for Enterprise Privacy Practices: Privacy-Enabled Management of Customer Data. In *Privacy Enhancing Technologies (PET)*, 2002.

[LAE+04] Kristen LeFevre, Rakesh Agrawal, Vuk Ercegovac, Raghu Ramakrishnan, Yirong Xu, and David J. DeWitt. Limiting Disclosure in Hippocratic Databases. In *Very Large Databases (VLDB)*, 2004.

[Lan01] Marc Langheinrich. A P3P Preference Exchange Language (APPEL1.0) Specification, 2001.

[LAP] Liberty Alliance Project, Privacy Preference Expression Languages (PPELs). http://projectliberty.org/liberty/content/download/371/2670/file/Final_PPEL_White_Paper.pdf.

[LAW] International Privacy Related Laws by Country and Region. http://www.informationshield.com/intprivacylaws.html.

[LCP+05] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Surveys and Tutorials*, 7(2), 2005.

[LMW02] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a Role-Based Trust-management Framework. In *IEEE Symposium on Security and Privacy (S&P)*, 2002.

[Luc10] Boris Lucas. Test de PriServ, un Service de Confidentialité de Données pour Environnements Pair-à-Pair. Technical report, Master Report, University of Nantes, 2010.

[MED] Global Physician Community Physician Connect. http://www.medscape.com/connect/.

[Mic61] James Michael. Justices of the Peace Act, 1361.

[MKL+02] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-Peer Computing. Technical report, HP Laboratories Palo Alto, 2002.

[MS03] Gerome Miklau and Dan Suciu. Controlling Access to Published Data Using Cryptography. In *Very Large Databases (VLDB)*, 2003.

[MUT] The MUTE Web Site. http://mute-net.sourceforge.net/.

[NAP] The Napster Web Site. http://free.napster.com/.

[NWQ+02] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. Edutella: A P2P Networking Infrastructure based on RDF. In *ACM World Wide Web Conference (WWW)*, 2002.

[OEC02] *OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data*. OECD Publishing, 2002.

[OLS] Oracle Label Security, http://www.oracle.com/us/products/database/label-security-066519.html.

[P3P] 1.0 P3P Purposes of Data Collection Elements. http://p3pwriter.com/LRN_041.asp.

[PEE] The PeerUnit Framework. http://peerunit.gforge.inria.fr/.

[PH09] Andreas Pfitzmann and Marit Hansen. A Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. Technical report, Dresden University of Technology, 2009.

[PSD02] Josep M. Pujol, Ramon Sangüesa, and Jordi Delgado. Extracting Reputation in Multi-Agent Systems by Means of Social Network Topology. In *Autonomous Agents and Multi-Agent Systems(AAMAS)*, 2002.

[RBTM07] Sunam Ryu, Kevin Butler, Patrick Traynor, and Patrick McDaniel. Leveraging Identity-Based Cryptography for Node ID Assignment in Structured P2P Systems. In *Advanced Information Networking and Applications Workshops (AINA)*, 2007.

[RD01] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *ACM/IFIP/USENIX Middleware Conference (MIDDLEWARE)*, 2001.

[RFSH01] Sylvia Ratnasamy, Paul Francis, Scott Shenker, and Mark Handley. A Scalable Content-Addressable Network. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, 2001.

[RH01] Antony Rowstron and George House. Storage Management and Caching in PAST, a Large-scale, Persistent Peer-to-Peer Storage Utility. In *Symposium on Operating Systems Principles (SOSP)*, 2001.

[RH03] Sandro Rafaeli and David Hutchison. A Survey of Key Management for Secure Group Communication. *ACM Computing Surveys*, 35, 2003.

[Riv90] Ronald L. Rivest. The MD4 Message Digest Algorithm. In *International Cryptology Conference (CRYPTO)*, 1990.

[Riv92a] Ronald L. Rivest. The MD5 Message-Digest Algorithm, 1992.

[Riv92b] Ronald L. Rivest. The RC4 Encryption Algorithm. Technical report, RSA Data Security, inc., 1992.

[RMI] The Java RMI. http://java.sun.com/.

[Rot00] Marc Rotenberg. Protecting Human Dignity in the Digital Age. Technical report, UNESCO, 2000.

[Rot01] Marc Rotenberg. Fair Information Practices and the Architecture of Privacy. *Stanford Technology Law Review*, 1, 2001.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2), 1978.

[RVLSA09] Claudia Roncancio, Maria Del Pilar Villamil, Cyril Labbé, and Patricia Serrano-Alvarado. Data Sharing in DHT based P2P systems. *Transactions on Large Scale Data and Knowledge Centered Systems*, 5740, 2009.

[San96] Ravi Sandhu. Role Hierarchies and Constraints for Lattice-Based Access Controls. In *European Symposium on Research in Computer Security*, 1996.

[SEL] SELinux, http://www.nsa.gov/research/selinux/index.shtml.

[SER] Sermo, http://www.sermo.com/.

[Ser06] Sergio Marti and Hector Garcia-Molina. Taxonomy of Trust: Categorizing P2P Reputation Systems. *Computer Networks*, 50, 2006.

[SFK00] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The NIST Model for Role-Based Access Control: Towards A Unified Standard. In *ACM Workshop on Role-based Access Control*, 2000.

[SHA95] *Secure Hash Standard 180-1.* U.S. Department of Commerce, National Institute of Standards and Technology, National Technical Information Service, 1995.

[SHA02] *Secure Hash Standard 180-2.* National Institute of Standards and Technology, 2002.

[SLB06] Rob Sherwood, Seungjoon Lee, and Bobby Bhattacharjee. Cooperative Peer Groups in NICE. *Computer Networks*, 50(4), 2006.

[SMK+01] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, 2001.

[SOL] The Oracle Solaris Web Site. http://www.oracle.com/solaris/.

[SOP] The SopCast Web Site. http://www.sopcast.org/.

[SQL] Microsoft SQL Server. http://www.microsoft.com/sqlserver/en/us/default.aspx.

[SS98] Pierangela Samarati and Latanya Sweeney. Protecting Privacy when Disclosing Information: k-Anonymity and its Enforcement through Generalization and Suppression. Technical report, Computer Science Laboratory, SRI International, 1998.

[SS01] Adam Stubblefield and Dan S.Wallach. Dagster: Censorship-Resistant Publishing without Replication. Technical report, Rice University, 2001.

[SS02] Jordi Sabater and Carles Sierra. Reputation and Social Network Analysis in Multi-Agent Systems. In *Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2002.

[ST04] Girish Suryanarayana and Richard N. Taylor. A Survey of Trust Management and Resource Discovery Technologies in Peer-to-Peer Applications. Technical report, University of California, Irvine, 2004.

[SW05] Ralf Steinmetz and Klaus Wehrle. Peer-to-Peer Systems and Applications. *Lecture Notes in Computer Science*, 3485, 2005.

[TIM+03] Igor Tatarinov, Zachary G. Ives, Jayant Madhavan, Alon Y. Halevy, Dan Suciu, Nilesh N. Dalvi, Xin Dong, Yana Kadiyska, Gerome Miklau, and Peter Mork. The Piazza Peer Data Management Project. *ACM Special Interest Group on Management Of Data (SIGMOD) Record*, 32(3), 2003.

[TRU] Medical Privacy Stories, Health Privacy Project. http://patientprivacyrights.org/media/True_Stories.pdf.

[USE] The Usenet news web site. http://usenet-news.net/.

[vC75] Entick v. Carrington. All English Law Repertory 41 (1558 to 1775), 1775.

[Vij10] Jaikumar Vijayan. P2P Networks a Treasure Trove of Leaked Health Care Data, Study Finds, 2010.

[WB90] Samuel Warren and Louis Brandeis. The Right to Privacy, 1890.

[Wes67] Alain F. Westin. Privacy and Freedom. In *Atheneum*, 1967.

[WM01] Marc Waldman and David Mazières. Tangler: A Censorship-resistant Publishing System based on Document Entanglements. In *Computer and Communications Security (CCS)*, 2001.

[WP02] Samuel M. Wilson and Leighton C. Peterson. The Anthropology of Online Communities. *Annual Review of Anthropology*, 31(1), 2002.

[XL04] Li Xiong and Ling Liu. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7), 2004.

[Yao03] Walt Yao. Fidelis: A Policy-Driven Trust Management Framework. In *Conference on Trust Management (iTrust)*, 2003.

[Yu01] Ting Yu. Interoperable Strategies in Automated Trust Negotiation. In *Computer and Communications Security (CCS)*, 2001.

[ZHS+04] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), 2004.

Appendix A

Résumé Étendu

La vie privée est un droit humain fondamental. Elle participe à la dignité humaine et d'autres valeurs telles que la liberté d'expression. Elle est devenue l'un des droits les plus importants de l'homme de l'âge moderne [Rot00].

A.1 Introduction

La reconnaissance de la vie privée est enracinée profondément dans l'histoire. D'abord elle est reconnue dans de nombreuses religions, civilisations et discussions philosophiques bien connues. Par exemple, dans la Grèce antique, Aristote a fait la distinction entre la sphère publique de l'activité politique et la sphère privée associé à la vie familiale et domestique. Les discussions à propos de la vie privée, dans l'histoire, ont toujours été préoccupées par la protection des informations personnelles de la divulgation dans les sociétés publiques.

Par la suite, les discussions sur la protection de la vie privée a émergé dans les pays occidentaux. Dans les années 1890, le futur président de la Cour suprême de justice des Etats-Unis, Louis Brandeis a formulé le concept de la vie privée qui fait valoir que l'individu a tout le droit "d'être laissé seul" [WB90]. Brandeis a soutenu que la vie privée est la plus chère des libertés dans une démocratie, et qu'elle devrait se refléter dans la Constitution. Edward Bloustein a dit que la vie privée est un intérêt humain qui protège la personnalité inviolable, l'indépendance de l'individu, la dignité et l'intégrité [Blo64].

De nos jours, l'intérêt pour le droit de la protection des informations personnelles a augmenté avec l'avènement des technologies de l'information. Le puissant potentiel de

surveillance des systèmes informatiques a incité les demandes pour des lois particulières qui régissent la collecte et le traitement des informations privées. La genèse d'une législation moderne dans ce domaine remonte à la première loi de protection des données dans le monde, adoptée en Allemagne en 1970. Cela a été suivi par des lois nationales en Suède (1973), les États-Unis (1974), Allemagne (1977) et en France (1978).

Alain Westin [Wes67] définit la confidentialité des données comme :

> *"le droit des individus de décider pour eux-mêmes* ***quand****,* ***comment*** *et* ***à quelle*** *mesure les informations leur concernant sont communiquées à d'autres."*

De cette définition, plusieurs questions peuvent être posées telles que :

- qui stocke les données?
- qui contrôle l'accès aux données?
- qui a accès aux données?
- comment les données sont-elles utilisées?
- pour combien de temps les données devraient-elles être stockées et utilisées?
- comment les propriétaires de données expriment-ils leurs préférences de confidentialité?
- comment les droits d'accès devraient-ils être obtenus?
- qui peut (ou devrait) revendiquer la propriété sur les données recueillies?

La confidentialité des données est traitée par de nombreuses organisations et législations qui ont défini des principes bien importantes. Dans la suite, nous présentons les législations de l'Organisation de Coopération et de Développement Économiques (OCDE) qui donnent une idée générale sur la relation entre la protection des données confidentielles et le domaine de droit. Beaucoup d'autres venus du monde entier sont répertoriés dans [LAW].

Directives de l'OCDE

L'Organisation de Coopération et de Développement Économiques (OCDE)[1], est l'une des plus grandes et les plus fiables sources de statistiques comparables au monde, sur les données économiques et sociales. Les directives de l'OCDE présentées dans [OEC02] déclare que:

- Il devrait y avoir des limites à la *collecte de données personnelles* et toutes les données doivent être obtenues par des moyens légaux et équitables et, le cas échéant, avec *le consentement* du *propriétaire des données* (paragraphe 7).

[1] http://www.oecd.org/.

- Les données personnelles doivent être pertinentes pour les *objectifs* pour lesquels elles doivent être utilisées, et, dans la mesure nécessaire à ces objectifs, les données doivent être exactes, complètes et tenues à jour (paragraphe 8).

- Les objectifs pour lesquels les données personnelles sont collectées, doivent être précisés au plus tard au *moment de la collecte de données* et par la suite *l'usage doit être limité* à la réalisation de ces objectifs ou d'autres qui ne sont pas incompatibles avec ces objectifs et qui sont précisés à chaque occasion de changement de finalité d'accès (paragraphe 9).

- Les données personnelles ne doivent pas être divulguées, ni utilisées pour des objectifs autres que ceux spécifiés conformément au paragraphe 9, sauf: a) avec le consentement de la personne concernée, ou b) par l'autorité de la loi (paragraphe 10).

En général, tous ces directives soulignent que la confidentialité des données doit tenir compte de: la limitation de la collecte des données, la spécification de l'objectif d'accès, la limitation de l'utilisation des données, la qualité des données, des mesures de sécurité, l'ouverture, la participation individuelle et la responsabilité. Tous ces principes doivent être respectés afin de protéger la confidentialité des données.

Nous verrons dans la suite que, même si ces lois protègent la confidentialité des données d'actions illégales des institutions et des entreprises, un simple utilisateur Web est capable de violer la confidentialité des données en raison de la divulgation des données et du partage.

Confidentialité et divulgation de données

La divulgation de données personnelles par les grandes entreprises a fait de la protection de la vie privée un besoin essentiel. La divulgation des données à caractère personnel, même par erreur, peut avoir des conséquences indésirables pour les propriétaires de données.

En Janvier 2000, une société qui vend des produits de santé en ligne, a révélé sur son site internet le nom de ses clients, leurs numéros de téléphone et même des informations sur leurs comptes bancaires et cartes de crédit, ce qui a causé une grande panique parmi ses clients.

Dans un autre exemple, pour des raisons pharmaceutiques, un service médical a communiqué sa base de données sur Internet. Elle comprenait les noms de 4,3 millions de personnes souffrant d'allergies, 923 000 ayant des problèmes urinaires, et 380 000 patients qui souffrent de dépression. Cela a eu des effets graves sur les personnes qui souhaitent conserver le caractère confidentiel de leurs données personnelles.

En Septembre 2003, après les accusations de viol contre le joueur de basket Kobe Bryant, les dossiers médicaux de la victime présumée ont été cités à comparaître dans un hôpital du Colorado par les avocats de la défense de Bryant. Après, un employé de l'hôpital a communiqué les documents au juge, les avocats de l'hôpital ont demandé au juge de rejeter les assignations et de détruire les dossiers déjà reçus par lui, en citant l'État

fédéral et les lois de confidentialité médicale. Les avocats de la victime ont également tenté d'empêcher l'équipe de défense de Bryant d'avoir accès à ces dossiers médicaux de deux autres hôpitaux. Toutefois, un certain nombre de journaux ont publié des informations médicales sensibles que les journalistes prétendent qu'ils sont venus des employés de l'hôpital. Dans ce cas, beaucoup de gens ont trouvé leurs informations médicales privées publiées au public intéressé par le cas Bryant.

Ces exemples et bien d'autres [TRU] nous donnent une idée sur les conséquences de la violation de la confidentialité à cause de la divulgation des données.

Confidentialité dans les communautés de partage en ligne

Pendant les quinze dernières années, la croissance mondiale de l'Internet a facilité l'émergence rapide d'interactions en ligne des communautés de personnes ayant des intérêts communs. Ces communautés en ligne présentent un large éventail de caractéristiques qui peuvent servir à diverses fins. Elles peuvent varier des petits groupes engagés dans des discussions ciblées sur des sujets précis, au mondes complexes créés par des centaines de participants simultanés. Maintenant, elles contiennent des millions d'utilisateurs liés par un intérêt dans les marchés ou les réseaux d'échange de biens et de l'information [WP02].

Les communautés en ligne telles que les communautés professionnelles (par exemple, médicale [MED, SER] ou les communautés de recherche [DOC]) gagnent en popularité en raison de besoins croissants sur le partage des données. Ils offrent souvent un soutien important pour les membres des communautés. Pourtant, afin de profiter de ce soutien, les utilisateurs doivent fréquemment divulguer des informations sensibles telles que les profils médicaux, les résultats de nouvelles recherches, etc.

La plupart des menaces à la vie privée au sein d'une communauté en ligne peuvent être classées en deux groupes distincts en fonction de l'objet de la divulgation: (1) les membres de la communauté affichent des informations privées sur quelqu'un d'autre et (2) les membres de la communauté affichent de renseignements personnelles sur eux-mêmes qui sont ensuite diffusées à l'extérieur de la communauté par un autre membre sans autorisation. Qu'advient-il lorsque d'autres membres de la communauté ne parviennent pas à garder l'information confidentielle potentiellement dangereuse? Les remèdes traditionnels ne seront probablement pas suffisantes pour protéger les personnes lorsque les membres d'une communauté de partage de données violent la confidentialité des autres membres. Dans ce contexte, la notion de *confiance* entre les membres pourrait jouer un grand rôle pour vérifier le comportement des utilisateurs et empêcher la violation de confidentialité.

Confidentialité dans les systèmes P2P

Les systèmes pair-à-pair (P2P) sont devenus une partie intégrante de l'Internet en attirant des millions d'utilisateurs. Selon des évaluations récentes [SW05], le trafic P2P dépasse désormais le trafic Web qui a été le trafic dominant sur Internet. Les applications P2P

les plus populaires restent le partage de fichiers et la distribution de contenu dans les communautés en ligne.

En quelques années, l'immense popularité des systèmes de P2P et l'explosion de la recherche dans le P2P ont créé un immense corpus de connaissances. Les environnements P2P pour les applications centrées sur les données offrent des caractéristiques intéressantes (par exemple, *la dynamicité, la distribution, l'autonomie*), mais des garanties limitées concernant la confidentialité des données. Ils peuvent être considérés comme hostiles, car les données, qui peuvent être sensibles ou confidentielles, peuvent être consultées par tous (par les pairs potentiellement malveillants) et utilisées pour tous (par exemple, pour le marketing, le profilage, la fraude ou pour des activités contre les préférences et l'éthique du propriétaire de données).

Une étude [GK03] a trouvé une douzaine d'exemples dans Kazaa [KAZ], où des informations sensibles comme des factures, des taxes et des adresses e-mails sont partagés avec insouciance dans la plupart des cas, mais malicieusement dans les autres cas. Kazaa expose des informations financières, des fichiers personnels et de la correspondance privée à des millions d'utilisateurs à travers le monde. Ces données pourraient être utilisées, par exemple, pour commettre une fraude (en particulier dans le monde de la finance), pour menacer la vie personnelle, ou même pour le vol d'identité.

Dans une étude plus récente [Vij10], professeur Eric Johnson, de Dartmouth College a décrit comment les chercheurs universitaires ont découvert des milliers de documents contenant des informations sensibles sur les patients dans les réseaux P2P populaires. Un des 3.000 fichiers découverts par les chercheurs a été une feuille de calcul contenant les détails d'assurance, des informations d'identification personnelle, des noms et des codes de diagnostic sur plus de 28.000 personnes. Un autre document contient des données similaires sur plus de 7.000 personnes. La plupart des documents contient des communications sensibles sur les patients, des données de traitement, des diagnostics médicaux et des évaluations psychiatriques. Au moins cinq dossiers contenaient suffisamment d'informations pour être considérés comme une violation majeure des règles actuelles en vertu de soins de santé.

Plusieurs systèmes P2P proposent des mécanismes pour assurer une sorte de protection des renseignements personnelles tels que OceanStore [**?**], Past [RH01], Freenet [CMH+02], etc. Toutefois, ces travaux restent insuffisants. Les lois sur la protection des données ont accentué le respect des préférences de confidentialité des utilisateurs. Ainsi, les systèmes P2P doivent prendre en compte les préférences de confidentialité ce qui n'est pas le cas dans les systèmes P2P actuels.

Le partage des données, avec des pairs de confiance, pour des objectifs et des opérations spécifiques, n'est pas possible sans l'ajout de nouveaux services. Dans ce contexte, un service de confidentialité efficace pour les systèmes P2P utiliant les objectifs d'accès et la confiance est nécessaire.

Objectif de la thèse

L'objectif de cette thèse est le suivant:

- Proposer des mécanismes pour la préservation et le contrôle de la vie privée qui peuvent contraindre les utilisateurs P2P de respecter les lois de confidentialité.
- Contribuer à l'élaboration d'un nouveau service efficace pour la confidentialité des données dans les systèmes P2P. Ce service doit fournir un environnement sûr pour le partage des données et le stockage dans les systèmes P2P, tout en préservant la confidentialité des données.
- Promouvoir l'utilisation des systèmes P2P entre les communautés professionnelles en ligne en appliquant un service pour empêcher la violation de la confidentialité des données.

Pour atteindre notre objectif, nous étudions des méthodes intéressantes et des techniques pour assurer la confidentialité des données. Nous examinons aussi les nombreux travaux qui utilisent ces techniques et nous les analysons afin de résumer les garanties nécessaires pour la protection de la confidentialité. Enfin nous proposons un modèle et un service de confidentialité pour répondre à ces besoins.

A.2 État de l'art

La fin du XXe siècle et les premières années du XXIe siècle a vu l'évolution rapide des télécommunications, des matériels et des logiciels. La croissance rapide et l'utilisation généralisée du traitement des données, réalisée par Internet, a alimenté le besoin de meilleures méthodes de protection des ordinateurs et des informations stockées, traitées et transmises. Les disciplines académiques de la sécurité informatique et de l'information ont émergé avec de nombreuses organisations professionnelles, tous partageant les objectifs communs d'assurer la sécurité et la fiabilité des systèmes d'information. Les principes de base de sécurité de l'information sont les suivantes: la confidentialité, l'intégrité et la disponibilité. Dans cette thèse, nous nous concentrons sur la confidentialité des données.

A.2.1 Techniques de confidentialité

Des recherches récentes ont été concentrées sur les techniques améliorant la vie privée et de proposer de nouveaux algorithmes pour préserver la confidentialité dans les systèmes d'information. Parmi eux, figurent:

- les techniques de contrôle d'accès, qui permettent de limiter l'accès non autorisé aux données privées. Les techniques de contrôle d'accès peuvent aider les utilisateurs à contrôler l'accès à leurs données privées. L'accès aux données peut être contrôlé à l'égard de l'identité de l'utilisateur, le rôle de l'utilisateur et l'objectif d'accès pour lequel les données des utilisateurs sont demandées.
- les techniques de cryptographie, qui sont les premières techniques utilisées pour assurer la sécurité des données, et restent actuellement les plus utilisées.

- les techniques d'anonymat, qui protègent la vie privée en rendant les utilisateurs indifférents des autres utilisateurs et à peine identifiés.

- les techniques de confiance, qui ajoutent de la protection de la confidentialité dans les systèmes d'information par la manipulation de la fiabilité des utilisateurs. Elles peuvent être utilisées pour prédire en quelque sorte le comportement des utilisateurs dans le système et donc de protéger la confidentialité des données contre les futurs actes malveillants.

- les bases de données Hippocratiques, qui intègrent une nouvelle notion de protection de la confidentialité, le contrôle d'accès basé sur les objectifs d'accès.

Nous supposons qu'aucune de ces techniques ne peut assurer seule la confidentialité des données dans des environnements ouverts. La combinaison de ces techniques fournissent plus de garanties de confidentialité. Par exemple, en assurant l'anonymat des utilisateurs, il est plus facile aux utilisateurs malveillants de violer la confidentialité des données, dans de telles circonstances, les techniques de confiance sont nécessaires. Parfois, il est très difficile de combiner des techniques. Par exemple, le contrôle d'accès qui a besoin de l'identité des utilisateurs ne peut pas être associé aux techniques de l'anonymat qui cachent l'identité des utilisateurs, dans ce cas, les cartes à puce ou les surnoms peuvent être utilisés.

A.2.2 Comparaison des systèmes P2P

Après avoir montré les techniques de confidentialité et les modèles qui peuvent être utilisés afin de protéger la confidentialité des données des systèmes d'information, nous montrons comment ces techniques sont utilisées et adaptées à des applications de partage de données dans les systèmes P2P.

Plusieurs systèmes P2P sont comparés sur la base des techniques utilisées pour protéger la confidentialité des données (cf. Tableau A.1) et les propriétés garanties de confidentialité (cf. Tableaux A.2 et A.3). Dans les tableaux de comparaison, une cellule est maintenu vide lorsque une technique de confidentialité (resp. une propriété) n'est pas utilisée (resp. garantie) par le système P2P. *N/A* est utilisé lorsque l'information sur une technique de protection des renseignements personnels ou une propriété n'est pas disponible dans la littérature.

Pour résumer, les techniques suivantes ont été étudiées dans la comparaison[2]:

- les techniques de contrôle d'accès qui sont en mesure d'empêcher l'accès non autorisé et malicieux.

- les techniques d'anonymat qui peuvent cacher les informations sur l'utilisateur et les données en assurant ainsi leur confidentialité.

[2]Nous n'avons pas trouvé dans la littérature un système P2P qui utilise les bases de données Hippocratiques, c'est pourquoi cette technique n'a pas été considéré dans la comparaison.

P2P Systems	Techniques de confidentialité				
	Contrôle d'accès	Anonymat	Confiance	Cryptage de données	Authentification des données
PAST	Utilisation des cartes à puces	Pseudonymat	Confiance basé sur les certificats		Utilisation des cartes à puces
OceanStore	Utilisation de deux types de restrictions (reader et writer)			Cryptage symétrique	Utilisation de hashage
Mnemosyne				Cryptage symétrique par block	
Office SharePoint Workspace	Utilisation d'espaces de travail et des listes de membres	Espaces de travail et groupes anonymes	Utilisation de couleurs de confiance	Cryptage symétrique	
Piazza	Utilisation des politiques d'accès définies par le propriétaire de données			Cryptage symétrique	
OneSwarm	Utilisation des permissions définies par le propriétaire de données	Communications anonymes	Utilisation des niveaux de confiance de communauté	Cryptage hybride	
Dagster		Communications anonymes		Cryptage symétrique par block	
Tangler		Communications et identités anonymes		Cryptage symétrique par block	
Freenet		Communications anonymes		Cryptage symétrique	Utilisation de clés de signatures
SwarmScreen		Communications aléatoires			

Table A.1: Comparaison des systèmes P2P en se basant sur les techniques de confidentialité utilisées

- les techniques de confiance qui gèrent le comportement des pairs et aident à distinguer les pairs malveillants de ceux dignes de confiance.
- Le cryptage des données et l'authentification qui protègent la confidentialité des données des lectures et écritures malveillantes, la corruption et la suppression.

En outre, les tableaux montrent comment les systèmes P2P peuvent garantir:

- la protection contre les lectures non autorisées en utilisant le cryptage des données.
- la protection contre la corruption et la suppression des données en utilisant des signatures des données.
- la divulgation limitée en utilisant des techniques de contrôle d'accès et de confiance.

Systèmes P2P	Propriétés garanties			
	Protection contre les lectures non autorisées	Protection contre la corruption et la suppression	Divulgation limitée	
			Propriétaire	Demandeur
PAST		Oui à cause de la signature des données	Oui grâce au contrôle d'accès (cartes à puce)	N/A
OceanStore		Oui à cause de la signature des données	Oui grâce au contrôle d'accès	N/A
Mnemosyne	Oui à cause du cryptage			
Office SharePoint Workspace	Oui à cause du cryptage		Oui grâce au contrôle d'accès	Oui grâce au contrôle d'accès
Piazza	Oui à cause du cryptage		Oui grâce au contrôle d'accès	Oui grâce au contrôle d'accès
OneSwarm	Oui à cause du cryptage		Oui grâce au contrôle d'accès	Oui grâce au contrôle d'accès
Dagster	Oui à cause du cryptage			
Tangler	Oui à cause du cryptage			
Freenet	Oui à cause du cryptage	Oui à cause de la signature des données		
SwarmScreen				

Table A.2: Comparaison des systèmes P2P en se basant sur les propriétés garanties

- les garanties d'anonymat et le démentie des liens des données en utilisant des techniques d'anonymat.
- La démenti de contenu des données en utilisant le cryptage des données.

La comparaison montre que les systèmes P2P n'utilisent pas complètement toutes les techniques de confidentialité, et aucun d'eux permet de garantir toutes les propriétés de confidentialité à la fois. Les principales raisons de ceci peuvent être résumées à des questions d'optimisation des coûts ou de différenciation dans les définitions pour la protection de la confidentialité.

Certains systèmes n'utilisent pas des techniques de confiance parce que:

- les techniques de confiance peuvent être très coûteuses ce qui peut avoir un impact sur le passage à l'échelle. Bien que ces systèmes privilégient la performance à la protection de confidentialité, ils sont vulnérables aux accès malveillants aux données et la confidentialité des données peut être facilement atteinte par les pairs malveillants. Trouver des techniques optimales de confiance qui introduisent des petits surcoûts peut être une solution pour ces systèmes.
- Certains systèmes comme Freenet, Dagster et Tangler préfèrent se concentrer sur des problèmes tels que le partage de données résistant à la censure ou l'anonymat

P2P Systems	Propriétés garanties				
	Anonymat			Démenti de liens	Démenti de contenu
	Auteurs	Serveurs	lecteurs		
PAST	Oui à cause de pseudonymat	Oui à cause de pseudonymat	Oui à cause de pseudonymat	Oui à cause de pseudonymat	
OceanStore					Oui à cause du cryptage
Mnemosyne					Oui à cause du cryptage
Office SharePoint Workspace	Oui à cause de l'anonymat d'espaces de travail	Oui à cause de l'anonymat d'espaces de travail	Oui à cause de l'anonymat d'espaces de travail	Oui à cause de l'anonymat d'espaces de travail	Oui à cause du cryptage
Piazza					Oui à cause du cryptage
OneSwarm	Seulement contre la surveillance par des tiers		Seulement contre la surveillance par des tiers	Oui à cause des communications anonymes	Oui à cause du cryptage
Dagster	Oui à cause des communications anonymes	Oui à cause des communications anonymes	Oui à cause des communications anonymes	Oui à cause des communications anonymes	Oui à cause du cryptage
Tangler	Oui à cause des identités anonymes	Oui à cause des identités anonymes	Oui à cause des identités anonymes	Oui à cause des communications anonymes	Oui à cause du cryptage
Freenet	Oui à cause des communications anonymes	Oui à cause des identités anonymes	Oui à cause des identités anonymes	Oui à cause des communications anonymes	Oui à cause du cryptage
SwarmScreen	Oui à cause des communications anonymes	Oui à cause des communications anonymes	Oui à cause des communications anonymes	Oui à cause des communications anonymes	

Table A.3: Comparaison des systèmes P2P en se basant sur les propriétés garanties - suite

des utilisateurs plutôt que de protéger la confidentialité des données des pairs malveillants.

Certains systèmes utilisent des techniques d'anonymat pour protéger la confidentialité des utilisateurs, mais la confidentialité des données n'est pas protégé parce que, avec des utilisateurs anonymes, les techniques de contrôle d'identité ne peuvent pas être employés. Ainsi, il n'est pas surprenant de voir que des systèmes comme Freenet, Dagster, Tangler et SwarmScreen, n'utilisent pas de techniques de contrôle d'accès car ils emploient des techniques d'anonymat. Ces systèmes sont vulnérables aux accès non autorisés. La confidentialité des données peut être facilement atteinte par les pairs malveillants qui ont toute la liberté d'accéder à toutes les données dont ils ont besoin.

A.3 PriMod: modéle de confidentialité pour les systèmes P2P

Le contrôle d'accès, la confiance, la cryptographie et d'autres techniques peuvent être combinés ensemble pour protéger la confidentialité des données dans les systèmes de gestion des données. Plusieurs approches ont été proposées pour protéger la confidentialité de données dans les systèmes P2P. Toutefois, si la notion d'objectif d'accès a une importance considérable, comme accentuée par les directives de l'OCDE et les principes hippocratiques, elle n'a pas encore été utilisée dans les systèmes P2P.

La nécessité de spécification de l'objectif d'accès motive la proposition d'un nouveau modèle de confidentialité basé sur l'objectif d'accès pour les systèmes P2P.

PriMod est un modèle de protection de données pour les systèmes de P2P qui combine le contrôle d'accès basé sur l'objectif, la confiance et des techniques cryptographiques. PriMod est proposé pour répondre à la demande de pairs de partager leurs données sensibles dans un système P2P, tout en préservant leur confidentialité.

Dans PriMod, les pairs sont des participants qui partagent des données, demandent des données, ou tout simplement contribuent à l'exécution d'un service ou d'une requête. Nous considérons qu'il y a des pairs qui possèdent des données et qui n'agissent pas nécessairement comme des serveurs de ces données. Ainsi, on distingue trois types de pairs:

- **Demandeur.** Un pair qui demande l'accès aux données.
- **Serveur.** Un pair qui stocke et fournit des données.
- **Propriétaire.** Un pair qui possède et partage les données.

PriMod ne fait aucune hypothèse sur l'organisation du système P2P qui peut être structuré, semi-structuré ou non structuré. L'unique hypothèse importante, qui doit être prise en compte par les pairs, est que chaque pair possède un identifiant unique dans le système pour toutes ses connexions. [CDG+02, RBTM07] ont traité de l'identification des pairs. Nous sommes pleinement conscients de l'impact de l'identification sur la vie privée des utilisateurs. Toutefois, les identités des pairs ne révèlent pas nécessairement les vrais identités des utilisateurs ainsi la confidentialité des utilisateurs peut être protégée.

Nous considérons que les pairs se conduisent mal si ils violent les préférences de confidentialité des données définies par les propriétaires des données. Dans ce modèle, nous nous concentrons sur la prévention des malveillances suivantes:

- **La divulgation non autorisée des données.** Les pairs serveurs peuvent se conduire mal en divulguant des données sensibles aux demandeurs qui, en se basant sur les préférences de la confidentialité des propriétaires, ne devraient pas avoir accès à ces données.
- **mal-utilisation de données.** Les pairs demandeurs se conduisent mal si ils ne respectent pas les accords conclus lors de l'accès aux données.

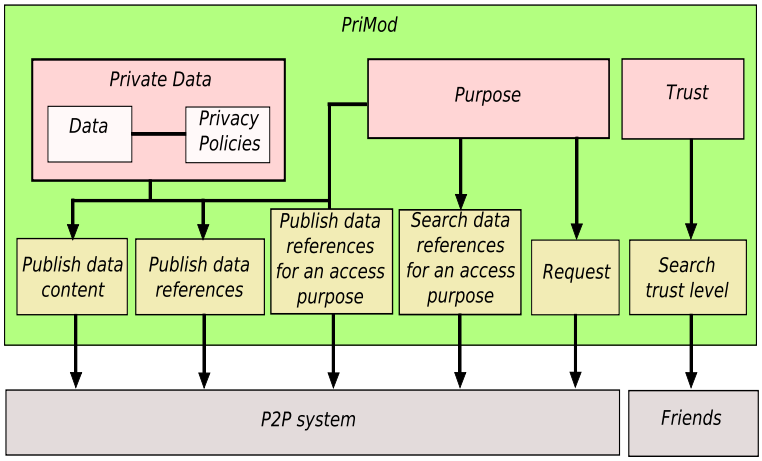


Figure A.1: PriMod

- **Les attaques à l'intégrité des données.** Les pairs Serveurs peuvent violer l'intégrité des données en modifiant, sans autorisation, le contenu des données qu'ils fournissent .

PriMod permet aux propriétaires de données de choisir entre la publication des références de leurs données (par exemple, les noms de fichiers, les clés primaires, etc) ou de publier le contenu des données chiffrées. Les pairs demandeurs peuvent rechercher des données sensibles, mais doivent préciser l'objectif d'accès et l'opération dans leurs demandes, donc ils s'engagent à l'usage prévu des données. PriMod permet également aux pairs demandeurs de savoir quelles données sensibles peuvent-ils accéder pour un objectif particulier.

Pour resumer, les propriétés et avantages de PriMod sont les suivants:

- Il bénéficie des avantages des systèmes P2P pour la publication et le partage des données tout en offrant plus de protection des données personnelles.
- Il peut être facilement intégré dans n'importe quel système P2P.
- Il propose des modèles pour definir les politiques de confidentialité, la confiance et des modèles de données, et offre des opérations pour la publication de contenu de données, la publication des références, la recherce de données, et la recherche des références pour un objectif d'accès particulier (cf. Figure A.1).
- Les propriétaires de données peuvent définir leurs préférences de confidentialité dans leurs politiques.
- Les données sensibles sont associées à des politiques de confidentialité plus précisément la notion d'objectif d'accès. Cette association crée les données privées qui peuvent être publiées dans le système.

- La recherche des données est toujours faite pour des objectifs et des opérations particuliers.
- Les techniques de confiance sont utilisés pour vérifier la fiabilité des pairs demandeurs.

En comparant PriMod aux modèles existants pour la protection de la confidentialité des données, nous constatons que:

- Comme de nombreux modèles, les données privées dans PriMod sont protégés contre les lectures malveillantes, les corruptions et les suppressions en utilisant le cryptage des données et les signatures. La divulgation de données est limitée en utilisant des techniques de contrôle d'accès.
- PriMod ne garantit pas l'anonymat car il n'est pas conçu pour protéger le comportement des utilisateurs, mais il se concentre sur la protection de la confidentialité de données.
- Contrairement à tous les modèles existants, PriMod comprend la notion d'objectifs d'accès en impliquant les principes Hippocratiques au modèle de données.

A.4 PriServ: Service de confidentialité pour les systèmes P2P

Un réseau pair à pair consiste simplement en un réseau connectant directement entre eux de nombreux participants (les pairs). Une application fonctionnant sur un réseau pair-à-pair est souvent orientée vers le partage de données, chacun pouvant publier ses données, demander des données, ou simplement héberger des données pour que d'autres puissent y accéder. Ces trois activités étant fondamentalement différentes, elles sont indépendantes au sein de PriServ. On parlera d'Owner (pair propriétaire de donnée), de requester (pair demandant une donnée) ou de Server (pair hébergeant et distribuant une donnée). Chaque pair peut jouer un ou plusieurs de ces rôles en même temps, selon qu'il publie, demande ou héberge une donnée.

Dans le cas de PriServ, on s'intéresse plus particulièrement aux DHT (Distributed Hash Table), comme Chord [SMK+01] et Pastry [RD01], qui permettent de construire des réseaux pair-à-pair structurés, de manière à retrouver une information associée à une certaine clé avec un nombre de message logarithmique par rapport au nombre de pairs.

Quand il s'agit d'un tel système, les attaques et actes malveillants peuvent se présenter sous de très nombreuses formes. C'est pourquoi PriServ n'est pas un mécanisme de sécurité général, prétendant empêcher toutes failles, mais plutôt un modèle permettant de prévenir contre des comportements précis qui sont spécifiés dans PriMod.

L'idée à PriServ est d'utiliser des politiques de confidentialité, définissant pour les données concernées différents critères d'accès à cette donnée.

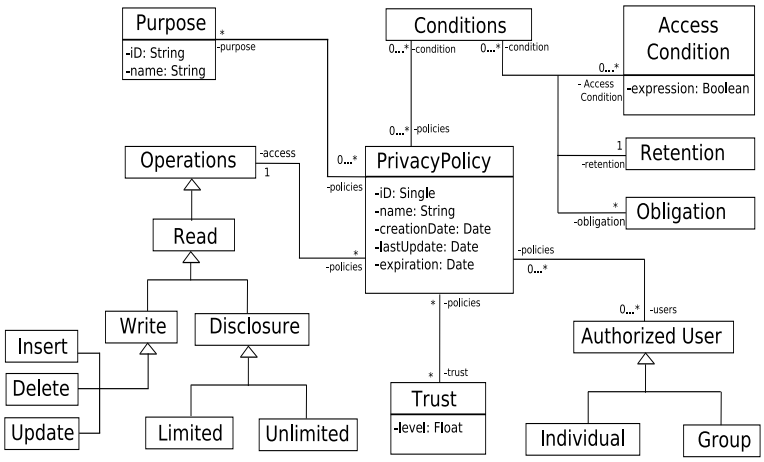


Figure A.2: Modèle de politique de confidentialité

D'un coté, on pourra autoriser l'accès à cette donnée pour des groupes d'utilisateurs, ou même pour certains utilisateurs spécifiques; mais on pourra également effectuer une sélection sur l'objectif de l'accès, et les opérations qui peuvent y être effectuées.

Rappelons également que PriServ n'est pas censé être une application à part entière, mais est plutôt destiné à être utilisé par une couche applicative, fournissant une interface utilisateur. PriServ est donc destiné à être libre, et est de plus destiné à être open source.

PriServ repose sur le principe des politiques de confidentialité. Ces politiques permettent de représenter des critères de confidentialité hétérogènes pour une même donnée. Voici les différents éléments composant celles utilisées dans PriServ, schématisé dans Figure A.2:

- **Authorized Users** (les utilisateurs autorisés). Une liste d'utilisateurs, ou de groupes d'utilisateurs, pouvant avoir accès à cette donnée.

- **Opérations**. Détermine ce qu'un utilisateur autorisé a le droit de faire avec la donnée. L'opération minimale est la possibilité de lire la donnée. On peut y ajouter des droits d'écriture, de distribution...

- **Purpose** (objectif). Spécifie pour quel(s) objectif(s) cette donnée peut être utilisée (e.g., but commercial, but scientifique uniquement,...).

- **Access Condition** (conditions d'accès). Spécifie des conditions sémantiques sur l'accès à la donnée.

- **Retention Time** (temps de rétention). Spécifie le temps durant lequel un utilisateur peut conserver la donnée.

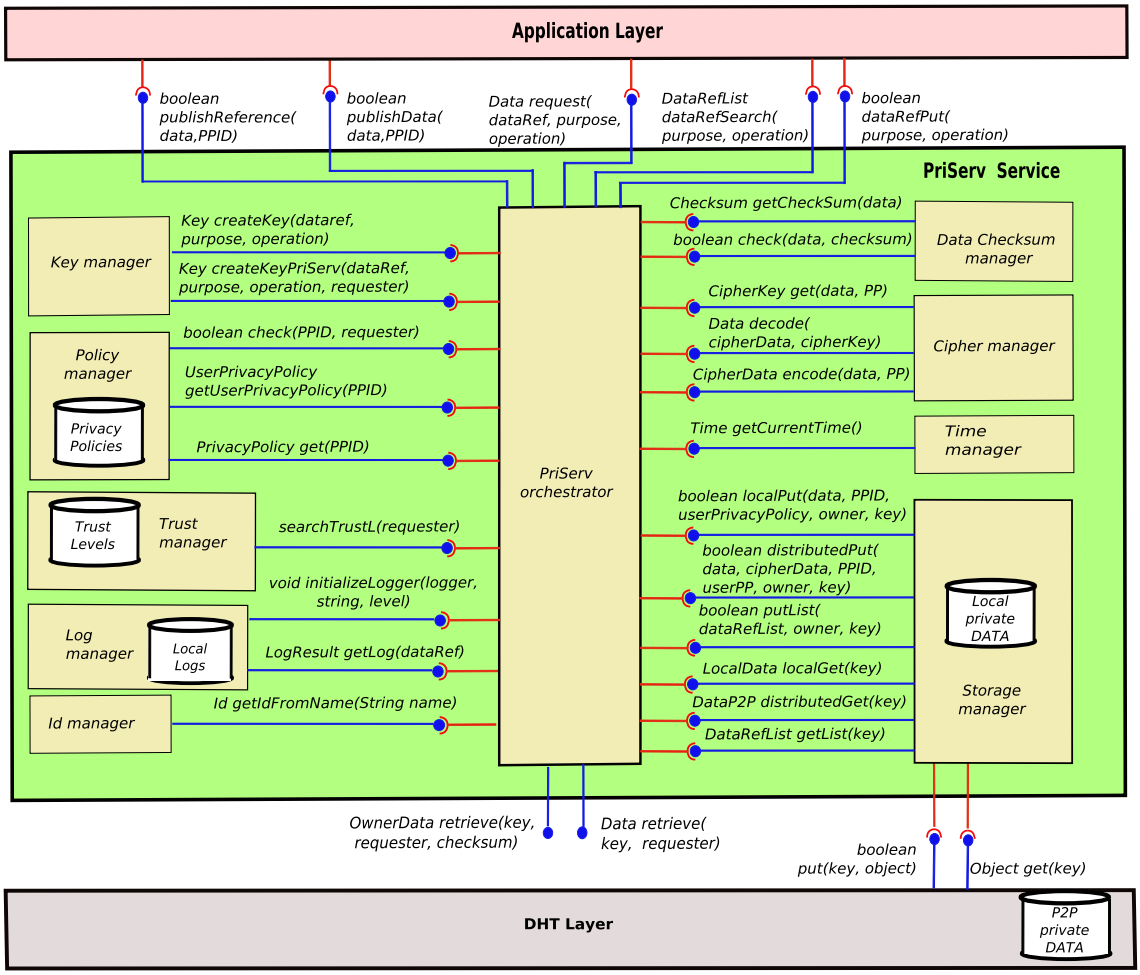


Figure A.3: Architecture des composants de PriServ

- **Obligations**. Spécifie les obligations auxquelles doivent se soumettre les utilisateurs après avoir eu accès à la donnée. Par exemple, avertir d'éventuels résultats de recherche dans le cas de données de recherches.
- **Minimal Trust Level** (niveau de confiance minimal). Spécifie le niveau de confiance minimal qu'un utilisateur doit avoir pour accéder à la donnée.

A.4.1 Architecture de PriServ

Afin de bien séparer les différents concepts vus précédemment (cryptage, génération de clé, gestion des politiques de confidentialité...), l'implémentation de PriServ est constituée de différents composants, offrant chacun des fonctionnalités précises.

Dans la Figure A.3, on peut voir l'ensemble des composants et leurs interactions.

1. **Key Manager** : Composant gérant les clés. C'est à lui qu'on fait appel pour créer

les clés permettant de stocker et récupérer des données sur la DHT, c'est aussi là qu'est défini l'interface KeyPriServ, étendant l'interface des clés de la DHT, qui permet d'y ajouter des informations utiles à PriServ (par exemple l'Id du requester).

2. **Cipher Manager** : Composant gérant le cryptage et décryptage des données. En plus de ces deux opérations, il conserve les clés de cryptage utilisées pour chaque donnée cryptée.

3. **Digest Manager** : Composant gérant les digest, servant à vérifier l'intégrité d'une donnée. Il est utilisé lorsqu'un requester demande une clé de décryptage, afin de vérifier que la donnée n'a pas été corrompue.

4. **Trust Manager** : Composant destiné à évaluer le niveau de confiance accordé à un pair donné.

5. **Policy Manager** : Composant dédié à la gestion des politiques de confidentialité, tant pour la création que pour le stockage. C'est également là que l'on trouve les implémentations des différentes entités définies plus haut.

6. **Storage Manager** : Composant de stockage. Un des composants principaux, puisqu'il gère aussi bien le stockage distant sur la DHT, dont il est le seul à avoir connaissance, que le stockage local des données, c'est à dire la table répertoriant les données stockées sur le réseau ainsi qu'une référence vers les politiques correspondantes.

7. **Id Manager** : Composant permettant d'identifier les autres pairs du réseau, à partir par exemple de l'URI qu'ils envoient, afin d'autoriser ou interdire l'accès à une donnée.

8. **Log Manager** : Composant de gestion des logs. S'occupe en particulier du logging en base de donnée, et de la récupération de logs, qui permettront l'audit du système.

9. **Time Manager** : Composant donnant le temps courant. Servira dans le cas où un mécanisme de synchronisation des horloges de tous les pairs connectés est mis en place, et donc où on ne se contente pas du temps système.

Ces trois derniers composants n'existaient pas dans la version de base de PriServ, et ont été ajoutés pendant le stage de Boris Lucas, pour répondre à des besoins inhérents à des fonctionnalités précises que nous voulions rajouter.

Tous ces composants sont liés par une même entité, appelée ici PriServ Orchestrator, qui va orchestrer l'utilisation des différentes interfaces proposées pour obtenir le comportement désiré. Il existe plusieurs type d'orchestrateurs, correspondant chacun à un des trois rôles déjà évoqué : l'owner, le requester, et le server.

L'orchestrateur server est en fait assez particulier, car il ne dispose en fait pas réellement d'interface destinée à une application externe, il se contente d'intercepter les get effectués sur la DHT, pour effectuer des vérifications particulières au mécanisme de PriServ auparavant. C'est ainsi qu'une requête pour une donnée sera refoulée au niveau du serveur si le demandeur ne fait pas partie des utilisateurs autorisés.

L'orchestrateur owner et requester fournissent les interfaces qui seront utilisées par la couche applicative pour stocker et récupérer des données. Les fonctions sont:

- publishReference(data, pcId). Publie une référence sur la donnée, les requesters devront demander directement la donnée (grâce à l'URI publiée).
- publishCryptedData(data, pcId). Publie la donnée cryptée, les requesters devront demander la clé de décryptage.
- requesting(dataRef, purpose, operation). Envoie une requête pour la donnée ayant cette référence *dataRef*, pour l'objectif *purpose* et l'opération *operation*.
- dataRefSearch(purpose). Envoie une requête pour les références des données accessibles pour l'objectif *purpose*.

A.5 Validation

PriServ a été simulé en utilisant SimJava [HM98]. L'analyse des coûts et de la simulation de PriServ révèlent que la publication de données dans le système conserve le coût logarithmique de la fonction put traditionnelle. Concernant les cout de la recherche, la recherche du niveau de confiance introduit un surcoût important, sinon les coûts sont logarithmique comme dans la fonction get traditionnelle. Toutefois, nous avons montré que le coût de recherche du niveau de confiance baisse avec le nombre de demandes et se stabilise.

Un prototype de PriServ est implémenté en Java en utilisant des outils SCA [Cha07] et RMI [RMI].

Les orchestrateurs vus précédemment forment le coeur de PriServ, en utilisant les différents composants mis à leur disposition. Pour assembler ces différents composants et expliciter leur agencement, la décision a été prise de se baser sur SCA (Service Components Architecture), qui est un ensemble de standards pour la création de services composites, et, le prototype de PriServ étant implémenté en java, d'utiliser la suite SCA Tools d'Eclipse. La suite SCA Tools propose un système simple d'annotations, permettant de déclarer des composants et des liens (binding) entre composants. Ainsi, chaque classe "façade" de composant est marquée à l'aide d'annotation, et chaque attribut faisant référence à un élément extérieur est également annoté. SCA Tools propose ensuite un éditeur graphique permettant d'assembler très simplement ces différents composants au sein d'un autre, en créant automatiquement des binding pour les références vers des éléments distants. Cet assemblage produit un fichier XML, qui, lors de l'exécution, sera chargé

pour instancier le composant désigné auparavant. Cette méthode permet de changer très facilement la configuration et l'agencement des différents composants, induisant ainsi une grande souplesse dans le développement d'application composites. Dans le cas de PriServ, qui rassemble de nombreux composants, on en comprendra l'utilité; de plus cela simplifie grandement la gestion de la communication avec la DHT, du moins au niveau de l'implémentation.

La conception et la mise en oeuvre de la DHT a été réalisée en collaboration avec Stéphane Drapeau de la société Obeo. Tous les composants ont été développés dans PriServ sauf les gestionnaires de journal, id et le temps qui ont été mis au point par Boris Lucas lors de son stage de recherche. Le prototype a été testé et validé sur Grid5000 en utilisant PeerUnit. Les résultats des tests nous a motivés pour construire une application pour la protection de confidentialité lors du partage des données médicales en utilisant le prototype PriServ.

Le site web PriServ est *http://atlas.lina.univ-nantes.fr/gdd/appa/priserv* et le code prototype peut être trouvé à *http://sourceforge.net/projects/priserv/*.

A.6 Conclusion

Pendant quelques années, les systèmes P2P ont acquis une grande popularité et ont attiré l'attention des applications de données en raison de leurs caractéristiques efficaces (par exemple, la dynamicité, la distribution, l'autonomie, etc.). Cependant, leurs garanties étant limitées concernant la confidentialité des données, ont découragé certaines applications avec des données sensibles de les utiliser. Les systèmes P2P pouvait être considéré comme une source de risques pour la confidentialité car, en principe, chaque pair dans le système a accès aux données partagées et il peut les utiliser pour des objectifs différents.

Dans ce contexte, l'objectif principal de cette thèse est de contribuer à l'élaboration d'un nouveau service efficace pour la protection de la confidentialité dans les systèmes P2P. D'autres objectifs étaient de promouvoir l'utilisation des systèmes P2P entre les applications de données en ajoutant des fonctionnalités pour la protection de la confidentialité, et d'obliger les utilisateurs P2P à respecter les lois de confidentialité.

Cette thèse présente une solution complète pour la protection de la confidentialité des données dans les systèmes P2P selon les préférences de confidentialité des utilisateurs.

Après avoir analysé les travaux existants sur les techniques de confidentialité et avoir comparé les systèmes P2P actuels proposé pour la protection de données personnelles, un modèle de confidentialité et un service de confidentialité pour les systèmes P2P ont été proposées et mises en oeuvre.

A.6.1 Contributions

État de l'art. La première contribution vise à une étude complète des techniques de confidentialité des données et des systèmes P2P, afin de motiver les prochaines

contributions. L'état de l'art a été réalisée en deux phases.

Tout d'abord, un sondage sur les techniques de protection de la confidentialité a permis d'identifier plusieurs façons de protéger les données personnelles: (a) les techniques de contrôle d'accès qui permettent de limiter les accès non autorisés aux données privées, (b) les bases de données Hippocratiques qui introduisent la notion d'objectif d'accès à la protection de la confidentialité, (c) les techniques d'anonymat qui protègent la confidentialité en rendant les utilisateurs indifférents des autres utilisateurs et à peine identifiés, (d) les techniques de confiance qui permettent de prédire le comportement des utilisateurs inconnus et donc offrir une protection contre les actes malveillantes, et (e) des techniques de cryptographie qui protègent la confidentialité des données en les rendant illisibles par les utilisateurs non autorisés.

Deuxièmement, une enquête et une comparaison ont été faites sur les services de confidentialité pour les systèmes P2P par rapport aux techniques utilisées pour protéger la confidentialité des données (contrôle d'accès, anonymat, confiance et cryptographie) et les propriétés de confidentialité garanties(la protection contre les lectures non autorisées, la corruption et suppression, la divulgation limitée, la garantie d'anonymat, le déni de contenu).

La première observation à partir de l'état de l'art a été que la combinaison des techniques de confidentialité augmente le niveau de protection. Cependant, certaines techniques ne peuvent pas être facilement combinées telles que les techniques d'anonymat qui cachent l'identité des utilisateurs, et les techniques de contrôle d'accès qui ont besoin de l'identité des utilisateurs pour limiter l'accès non autorisé. La deuxième observation est que les systèmes P2P n'ont pas utilisé complètement toutes les techniques de confidentialité, et aucun d'eux garantit toutes les propriétés de confidentialité à la fois. La troisième observation est que, même si la notion d'objectifs d'accès a une importance considérable, comme accentuée par les directives de l'OCDE et les principes Hippocratique, elle n'a pas encore été utilisée dans les systèmes P2P. Ainsi, une nouvelle solution pour la protection de confidentialité dans les systèmes P2P qui utilise le contrôle d'accès basé sur l'objectif d'accès est nécessaire.

PriMod Motivé par la nécessité de spécification de l'objectif d'accès, la deuxième contribution de ce travail vise à proposer un nouveau modèle de confidentialité pour les systèmes P2P. PriMod a été proposé pour bénéficier de l'efficacité de P2P pour publier et partager des données tout en protégeant la confidentialité des données. PriMod propose un modèle de politique de confidentialité qui permet aux propriétaires de définir leurs préférences de confidentialité dans des politiques, un modèle de confiance pour prédire le comportement des pairs, et un modèle de données dans lequel les données sensibles sont associés à des politiques de confidentialité. PriMod offre également des opérations pour la publication de contenu de données, la publication des références, la recherche des données sensibles, et la recherche des références de données autorisées pour un objectif particulier. Les recherches sont toujours faites pour des objectifs et des opérations particuliers ce qui engage les demandeurs à l'usage prévue de données.

PriMod protège les données sensibles contre les lectures malveillantes, les corruptions

et les suppressions à l'aide de cryptage et de signatures des données. La divulgation des données est limitée en utilisant des techniques de contrôle d'accès. PriMod ne garantit pas l'anonymat car il n'est pas conçu pour protéger le comportement des utilisateurs, mais se concentre sur la protection des données. Contrairement à tous les modèles trouvés dans la littérature, PriMod comprend la notion de l'objectif d'accès en impliquant les principes Hippocratiques au modèle de données P2P.

PriServ La troisième contribution vise à construire un service de protection des données privés sur des systèmes P2P basé sur les DHTs. En se basant sur PriMod, PriServ empêche la violation de la confidentialité en limitant les accès malveillants aux données. Il combine le contrôle d'accès à base d'objectifs avec les techniques de confiance, les techniques de cryptographie et les signatures numériques et propose des fonctions pour la publication et la recherche de données en préservant leurs confidentialité. PriServ peut être exécuté sur n'importe quel système DHT tant qu'il précise les méthodes put et get permettant de publier et obtenir des données du système. La publication et la recherche de données dans PriServ sont toujours faites pour des objectifs et des opérations spécifiques. Cela engage les demandeurs d'utiliser les données uniquement pour des fins prévues et à effectuer des opérations bien spécifiées. Légalement, cet engagement peut être utilisé contre les demandeurs malveillants s'il est avéré que les données obtenues ont été utilisées différemment.

PriServ introduit un surcoût acceptable sur les systèmes de DHT. L'analyse des coûts et la simulation ont révélé que les coûts de publication et de recherche sont logarithmiques comme dans les DHT traditionnels. Toutefois, la recherche du niveau de confiance dans PriServ introduit un surcoût linéaire, ce qui demande des activités de recherche supplémentaires dans ce domaine.

Initialement, les fonctionnalités de PriServ ont été simulées afin de valider les coûts de ces algorithmes. Ensuite, un prototype de PriServ a été implémenté et mis en oeuvre. Ce prototype a été testé et validé par PeerUnit sur Grid5000 et a été utilisé pour construire une application de partage de données sensibles pour les environnements médicaux.

A.6.2 Travaux futurs

Les travaux futurs se concentrent sur l'extension de PriServ avec des fonctionnalités plus avancées et en l'étendant à d'autres contextes. Cette section présente une liste non exhaustive des travaux qui seront la continuité de cette thèse.

Vérification de la conformité Afin de protéger la confidentialité des données, il est nécessaire, non seulement de permettre aux propriétaires de données de spécifier la manière dont leurs données sensibles doivent être accessibles, mais aussi de fournir des outils d'audit, qui seront utilisés pour vérifier le respect des préférences de confidentialité spécifiées. L'architecture de PriServ contient un gestionnaire de journaux qui gère un journal pour chaque pair dans le système. Chaque action concernant le partage de données sensible est inscrite dans le journal correspondant. Les travaux futurs se concentrent sur

la conception et la construction d'un service d'audit pour PriServ qui utilise les journaux des pairs pour former des requêtes d'audit. L'idée est d'ajouter un gestionnaire de la vérification à l'architecture de PriServ qui sera en charge de la collecte d'informations à écrire dans les enregistrements du journal et, plus tard, pour être en mesure de les récupérer.

Groupes de demandeurs et gestion de la confiance Dans PriServ, lorsque le contenu de données crypté est publié, les propriétaires doivent être contactés par les demandeurs afin de récupérer les clés de déchiffrement symétrique. Une amélioration de PriServ est d'éviter les goulets d'étranglement sur les propriétaires afin d'avoir un meilleur passage à l'échelle. Pour cela, les clés asymétriques peuvent être utilisés pour le cryptage du contenu de données ce qui évite l'échange de clés symétriques. Afin d'éviter d'avoir un grand nombre de clés asymétriques générées, les demandeurs pourraient être organisés en groupes et chaque groupe se verra attribué une paire de clés publique/privée.

La gestion de la confiance pourrait également être faite par groupe. Un demandeur doit avoir un niveau de confiance minimal pour faire partie d'un groupe. Si un demandeur est dans un groupe, il peut accéder aux données sensibles autorisées pour ce groupe. Quand un demandeur se conduit mal, son niveau de confiance sera décrémenté. Lorsque le niveau de confiance du demandeur devient inférieur au niveau de confiance minimal d'un groupe, les membres du groupe voteront pour éjecter ce demandeur de leur groupe.

Cette proposition conduit à d'autres questions de recherche (par exemple, la définition et l'entretien du groupe, la mise à jour du niveau de confiance, le consensus du groupe, etc.) qui nécessitent des études plus approfondies et font parties des travaux futurs.

P2PWeb L'objectif du projet P2PWeb est de concevoir et développer un système de distribution de contenu multimédia dans un environnement Web en utilisant des techniques P2P. Ce projet vise à étudier les technologies P2P afin de construire un système qui permet aux utilisateurs de récupérer, via leur Internet Explorer, le contenu multimédia non seulement à partir du serveur fournisseur de contenu, mais aussi des utilisateurs qui ont déjà ce contenu. Dans ce contexte, le contrôle de confidentialité et la gestion de la satisfaction sont essentiels ce qui constitue un axe majeur du projet. Les contributions de cette thèse (l'état de l'art, PriMod, PriServ et son prototype) peuvent donner une base solide pour proposer des solutions pour la protection de la confidentialité du contenu multimédia dans des environnements P2P.

Data Privacy in P2P System

Mohamed Jawad

Abstract

Online peer-to-peer (P2P) communities such as professional ones (e.g., medical or research communities) are becoming popular due to increasing needs on data sharing. P2P environments offer valuable characteristics but limited guarantees when sharing sensitive data. They can be considered as hostile because data can be accessed by everyone (by potentially malicious peers) and used for everything (e.g., for marketing or for activities against the owner's preferences or ethics). This thesis proposes a privacy service that allows sharing sensitive data in P2P systems while protecting their privacy. The first contribution consists on analyzing existing techniques for data privacy in P2P architectures. The second contribution is a privacy model for P2P systems named PriMod which allows data owners to specify their privacy preferences in privacy policies and to associate them with their data. The third contribution is the development of PriServ, a privacy service located on top of DHT-based P2P systems which implements PriMod to prevent data privacy violations. Among others, PriServ uses trust techniques to predict peers behavior.

Keywords: Data Privacy, Purpose-based Access Control, Trust, P2P systems, distributed hash tables, Hippocratic databases.

Confidentialité des Données dans les Systèmes P2P

Résumé

Les communautés en ligne pair-a-pair (P2P), comme les communautés professionnelles (p. ex., médicales ou de recherche) deviennent de plus en plus populaires a cause de l'augmentation des besoins du partage de données. Alors que les environnements P2P offrent des caractéristiques intéressantes (p. ex., passage a l'échelle, disponibilité, dynamicité), leurs garanties en termes de protection des données sensibles sont limitées. Ils peuvent être considérés comme hostiles car les données publiées peuvent être consultées par tous les pairs (potentiellement malicieux) et utilisées pour tout (p. ex., pour le commerce illicite ou tout simplement pour des activités contre les préférences personnelles ou éthiques du propriétaire des données). Cette thèse propose un service qui permet le partage de données sensibles dans les systèmes P2P, tout en assurant leur confidentialité. La première contribution est l'analyse des techniques existant pour la confidentialité de données dans les architectures P2P. La deuxième contribution est un modèle de confidentialité, nomme PriMod, qui permet aux propriétaires de données de spécifier leurs préférences de confidentialité dans de politiques de confidentialité et d'attacher ces politiques a leurs données sensibles. La troisième contribution est le développement de PriServ, un service de confidentialité, base sur une DHT qui met en oeuvre PriMod afin de prévenir la violation de la confidentialité de données. Entre autres, PriServ utilise de techniques de confiance pour prédire le comportement des pairs.

Mots-clés : Confidentialité de données, Objectif d'Accès, Confiance, Système Pair-à-Pair, DHT, Base de Données Hippocratiques.

Discipline : Informatique
Spécialité : Bases de Données

Laboratoire : Laboratoire d'Informatique de Nantes Atlantique.
umr cnrs 6241. 2, rue de la Houssinière, bp 92 208 – 44 322 Nantes, Cedex 3.