Technical University of Catalonia

PhD Dissertation

Contributions to security and privacy protection in recommendation systems

Author:
Juan Vera del Campo

Advisor:
Josep Pegueroles Vallés

Telematics Engineering Department

August 2012

Barcelona tower, inbound for landing. Echo Charlie Lima Oscar Bravo.

Abstract

This thesis explores the problem of finding new and interesting documents in distributed networks using recommender systems. A recommender system is defined as an automatic system that, given a customer model and a set of available documents, is able to select and offer those documents that are more interesting to the customer. We will establish that the desirable characteristics of a recommender system are (i) to be fast, (ii) distributed and (iii) secure. A fast recommender system enhances the shopping experience of the client, since a recommendation is not useful if it arrives too late. A distributed recommender system prevents the creation of centralized databases with sensitive information and improves the availability of the documents. Finally, a secure recommender system protects every participant of the system: users, content providers, recommenders and intermediate nodes.

From the point of view of security, there are two main issues that recommender systems must face: (i) protection of the users' privacy and (ii) protection of all participants in the recommendation process. Recommenders issue personalized recommendations taking into account not only the profile of the documents, but also the private information that customers send to the recommender. Hence, the users' profiles include personal and highly sensitive information, such as their likes and dislikes. The second challenge that recommender systems face involves a new kind of attack. New legislation trends such as ACTA, SOPA or the "Sinde-Wert law" in Spain show the interest of states all over the world to control and prosecute nodes that aid in the process of discovering and getting copyrighted documents. Recent trials such as MegaUpload, PirateBay or the case against Mr. Pablo Soto in Spain show that these threats are a reality.

To achieve these goals, during the development of this thesis we propose the next contributions:

1. A social model that captures user's interests, and a metric function that calculates the similarity between users, queries and documents. This model represents profiles as vectors of a social space. Document profiles are created by means of the inspection of the contents of the document. Then, user profiles are calculated as an aggregation of the profiles of the documents that the user owns. Finally, queries are a constrained view of a user profile. This way, all profiles are contained in the same social space, and the similarity metric can be used on any pair of them.

2. Two mechanisms to protect the personal information in the users' profiles. The first mechanism takes advantage of the Johnson-Lindestrauss and Undecomposability of random matrices theorems to project profiles into social spaces of less dimensions. Even if the information about the user is reduced in the projected social space, we will show that under certain circumstances the distances between the original profiles are maintained. The second approach uses a zero-knowledge protocol to answer the question of whether or not two profiles are affine without leaking any information in case of that they are not.

3. A distributed system named DocCloud that protects merchants, customers and indexers against legal attacks, by means of providing plausible deniability and oblivious routing to all the participants of the system. DocCloud organizes databases in a tree-shape structure over a cloud system and provide a Private Block Retrieval protocol to avoid that any participant or observer of the process can identify the recommender. This way, customers, intermediate nodes and even databases are not aware of the specific database that answered the query.

4. A social, P2P network where users link according to their similarity, clustering and randomness, and provide recommendations to other users in their neighborhood. In order to create this social network, we define an epidemic protocol to discover affine users and establish other additional mechanisms that speed up the creation of the social network.

5. A distributed filesystem that provides documents. In our view of a recommender system, a recommendation is a complete process that ends when the customer receives the recommended document. We develop a distributed and secure filesystem where merchants are protected and cannot be identified, documents are private and authorization is required to access them, and it includes extra mechanisms to enhance the availability of the documents.

Resumen

Este documento explora cómo localizar documentos interesantes para el usuario en grandes redes distribuidas mediante el uso de sistemas de recomendación. Se define un sistema de recomendación como un sistema automático que, dado un modelo de cliente y un conjunto de documentos disponibles, es capaz de seleccionar y ofrecer los documentos que son más interesantes para el cliente. Las características deseables de un sistema de recomendación son: (i) ser rápido, (ii) distribuido y (iii) seguro. Un sistema de recomendación rápido mejora la experiencia de compra del cliente, ya que una recomendación no es útil si es que llega demasiado tarde. Un sistema de recomendación distribuido evita la creación de bases de datos centralizadas con información sensible y mejora la disponibilidad de los documentos. Por último, un sistema de recomendación seguro protege a todos los participantes del sistema: usuarios, proveedores de contenido, recomendadores y nodos intermedios.

Desde el punto de vista de la seguridad, existen dos problemas principales a los que se deben enfrentar los sistemas de recomendación: (i) la protección de la intimidad de los usuarios y (ii) la protección de los demás participantes del proceso de recomendación. Los recomendadores son capaces de emitir recomendaciones personalizadas teniendo en cuenta no sólo el perfil de los documentos, sino también a la información privada que los clientes envían al recomendador. Por tanto, los perfiles de usuario incluyen información personal y altamente sensible, como sus gustos y fobias. Con el fin de desarrollar un sistema de recomendación útil y mejorar su eficacia, creemos que los usuarios no deben tener miedo a la hora de expresar sus preferencias. Para ello, la información personal que está incluida en los perfiles de usuario debe ser protegida y la privacidad del usuario garantizada.

El segundo desafío desde el punto de vista de la seguridad implica un nuevo tipo de ataque. Dado que la prevención de la distribución ilegal de documentos con derechos de autor por medio de soluciones técnicas no ha sido eficaz, los titulares de derechos de autor cambiaron sus objetivos para atacar a los proveedores de documentos y cualquier otro participante que ayude en el proceso de distribución de documentos. Además, tratados y leyes como ACTA, la ley SOPA de EEUU o la ley “Sinde-Wert” en España ponen de manifiesto el interés de los estados de todo el mundo para controlar y procesar a estos nodos intermedios. Los juicios recientes como MegaUpload, PirateBay o el caso contra el Sr. Pablo Soto en España muestran que estas amenazas son una realidad.

Para alcanzar estos objetivos, durante el desarrollo de esta tesis se han realizado las siguientes contribuciones:

1. Un modelo social que capte los intereses del usuario, y una métrica que calcule la similitud y afinidad entre usuarios, consultas y documentos. Este modelo representa los perfiles como vectores de un espacio social. Los perfiles de documento se crean por medio de inspeccionar el contenido del documento mientras que los perfiles de usuario se calculan como una agregación de los perfiles de los documentos que posee el usuario. Por último, las consultas se definen como una vista limitada de un perfil de usuario. De esta manera, todos los perfiles están contenidos en el

mismo espacio social, y la métrica social de similitud definida se puede utilizar con cualquier par de ellos.

2. Dos mecanismos para proteger la información personal que contienen los perfiles de usuario. El primer mecanismo utiliza los teoremas de Johnson-Lindestrauss y descomposición de matrices aleatorias para proyectar los perfiles en espacios sociales de menos dimensiones. Incluso si la información que el perfil contiene sobre el usuario se reduce al proyectar en el nuevo espacio social, mostraremos que bajo ciertas condiciones las distancias entre los perfiles proyectados se mantienen. El segundo enfoque utiliza un protocolo de conocimiento cero para responder a la pregunta de si dos perfiles son afines o no, sin descubrimiento de cualquier información en caso de que no lo sean.

3. Un sistema distribuido llamado DocCloud que protege a los comerciantes, clientes e indexadores contra los ataques legales, mediante el suministro de negación a todos los participantes del sistema. DocCloud organiza las bases de datos e indexadores en una estructura en forma de árbol que se monta sobre un sistema cloud-computing y proporciona un protocolo de Private Block Retrieval para evitar que cualquier participante u observador del proceso puede identificar el recomendador. De esta manera, los clientes, los nodos intermedios e incluso los propios recomendadores no saben quién ha respondido una petición de recomendación.

4. Una red social P2P donde los usuarios se enlazan de acuerdo a su semejanza, y ofrecen recomendaciones a otros usuarios en su vecindario. Se define un protocolo de enrutamiento epidémico donde los enlaces se establecen de acuerdo a la similitud con los vecinos vecinos, el grado de agrupación y el azar. Además, se proponen una serie de mecanismos adicionales como el SoftDHT, que ayudan en la identificación de los usuarios afines y aceleraran el proceso de creación de grupos de usuarios similares.

5. Un sistema de distribución de documentos que proporciona los documentos recomendados al final del proceso. En nuestra opinión, un sistema de recomendación debe ser un proceso completo que acaba cuando el cliente recibe finalmente el documento recomendado. Se propone SCFS, un sistema de archivos distribuido y seguro donde los comerciantes están protegidos y no pueden ser identificados, los documentos son privados y se requiere autorización para acceder a ellos, y que incluye mecanismos adicionales para mejorar la disponibilidad de los documentos.

Acknowledgments

This document and the process that took to this moment would have been absolutely impossible without the patience and constant support from my advisor, Josep Pegueroles. To him, my humble acknowledgments and admiration. Any additional word that I may use to show my gratitude for his efforts would be necessarily a gray shadow of my true feelings.

Only second to Josep, I must thank the help and support during these years from Juan Hernández-Serrano. I have learned a lot from him and his constant enthusiasm. You rule, man. Of course, I must extend without hesitation this gratitude to all the outstanding members of the Information Security Group of the Telematics Engineering Department of the UPC. I enjoyed a lot my long membership to this group.

Next, I'd like to show my gratitude the COSIC research group that invited me for no less than two short stays at the KU Leuven, Belgium. All members of COSIC have something to teach, and I learned a lot at their premises. Their influence upon some parts of this document should be evident.

Last but not least, I'd like to show my especial gratitude and love to Astrid and her patience during these long nights and weekends. I'm still a bit surprised she survived the process. Moltes besaes, mante.

Contents

List of Figures

List of Tables

Part I.

Introduction

Part I: Introduction

Making use of recommendations during commercial transactions is older than commerce. Probably, a couple of minutes before that the first trade took place, some words about the advantages of the product were exchanged: “try these gorgeous, delicious, tasty Brussels sprouts I grew in my garden. I’ll give them to you if you give me two mammoths”. This was a step forward from the traditional “I collect my own wild fruits, I hunt my own mammoths”. Today, we are no longer interested in mammoths (unfortunately, we still eat sprouts) but we do use similar recommendations during our every day commercial transactions. Our clever caveman had the excellent idea of praising his useless sprouts for getting rid of them. After all, those sprouts started the first social wave that Alvin Toffler described in his seminal work [126]. Recommend exactly the product that the seller had to sell was an excellent and highly profitable recommender system for the next millennia, regardless of the real interest of the costumer.

In the early twentieth century, Henry Ford rode the second of the Toffler’s waves while stating his famous sentence: “any customer can have a car painted any color that he wants so long as it is black”. Uniformed and standardized mass production was born. In any case, since every car was virtually the same that the next one, we still needed a crafted seller/recommender to decide which was the most suitable car for our needs. Curiously enough, it was always the most expensive in the shop, but we were able to decide the color regardless of Mr. Ford’s thoughts.

Toffler foresaw the come of the third social wave, an era of *prosumers*[1] and highly personalized products. “Demassification”, “diversity”, “knowledge-based production” were going to be trend topics in the new society. According to Toffler, costumers won’t visit the mall anymore; they will satisfy their own needs using concepts that are very similar to “open source” and “Ikea”. Products were going to be no longer standardized, and prosumers will have the chance of personalize them to suit their needs.

Toffler was wrong. We are still going to the market. However, it is now hundreds of thousands of square kilometers wide. It is called *the Internet* and it enables sellers to advertise their products from any corner of the world using only a computer. Products are not personalized, but the number of different ones is so overwhelmingly high that they *seem* personalized. There is no need to waste time personalizing something when there is someone somewhere that makes exactly what we want and is willing to send it

[1]prosumer=producer+consumer

to us for a small fee. Every time that we switch our device on, a wave made of millions of products, documents and web sites is about to fall all over our heads.

Some voices alert that current supply is so high that we will never be able to consume all of it, being the basis of the current economic crisis and the main challenge that the developed economies will have to face during the next decade. We commoners are not concerned about these issues yet, but we really need a system to tidy up this mess, a system that surfs the wave of products for us looking for interesting things, avoids the Brussels sprouts and comes back with our beloved, long missed mammoths.

This is a work for an automatic recommender system. And we want it to be secure.

1. Recommender systems

Given the huge amount of information that is available on the Internet about products and costumers, is it possible to offer a recommender systems that provide *better* recommendations than humans?

Twenty years have passed since the software systems that are now known as *recommender systems* were first developed [50, 106]. Since that moment, researchers have continuously developed new approaches for implementing recommender systems, and today most of us are used to receive support to our commercial decisions from recommendation services such as the one used by Amazon. The application area of recommender systems is huge and commercially extensive, providing solutions in domains as diverse as financial products, real estate, electronic consumer products, movies, books, music, news, and web sites. An early commercial recommender systems survey can be found in [112], and some of the analyzed systems are still in use.

In this thesis, we will look to the recommender system from the perspectives of networking and security. Thus, for the purposes of this research, a *better* recommendation is achieved if (i) the output of a recommendation arrives immediately; (ii) the user and other participants of the network can get protection against people collecting user's profiles and powerful lawyers willing to avoid exchange of digital content; and (iii) the mechanisms to provide features (i) and (ii) do not affect the correctness of the recommendations.

Many challenges await in the way of designing such a recommender system. For example, users are not rational, they do not know what they want, or find difficulties in explaining what they want. The way that an interviewer takes to ask for the users' profile may affect their queries to the system, and result in unsatisfactory recommendations. Sometimes, there is a high enter barrier to a recommender system, pushing users to fill lengthy surveys about their interests. Getting useful recommendations fast is a desirable characteristic of the system. In addition, most recommender systems construct extremely detailed users' profiles that contain a great amount of personal information. This information is stored in a centralized server that users have to trust, or is spread in a decentralized network of uncontrolled nodes. Finally, the source of a recommendation or the final provision of the desired product may face copyright infringements. We believe that a *better* recommender system must face all these challenges. Unfortunately, this is not the case of most successful recommender systems in use today, as we will see during this chapter.

We focus our efforts in defining a fast, distributed and secure recommender system. These three parameters, *fast*, *distributed* and *secure*, will set bounds to the available mechanisms to describe users and classify products. Hence, even if we won't focus our efforts on defining a metric to capture whether or not a product is interesting, the requirements of the proposed recommender system would limit the number of available metrics.

Fast recommender system. Recommendations and e-commerce are tightly related fields. A potentially useful recommendation becomes useless if it arrives too late. The shopping experience is enhanced if recommendations are both accurate and fast. We assume accuracy, and this is a safe assumption since commercial recommender systems achieve a very acceptable degree of success. In a distributed environment such as the Internet, there are millions of potential clients, tons of available resources and there are too many different users' interests. Getting a fast recommendation is a hard problem if users are distributed all over the world as a decentralized system and can search resources using a customized query.

Distributed recommender system. Centralized recommender systems are extensively used in current electronic commerce. However, even if some techniques could be used in centralized systems to protect user's privacy (as distortion of the user's profiles before inserting them into a centralized database, or anonymous queries), distributed recommender systems are considered more secure because they prevent the creation of a single database with all information and avoid usual centralized security attacks. Furthermore, the deployment of a distributed system to serve millions of users is faster and cheaper than any centralized service. In this distributed scenario, as we will see next, old privacy preserving techniques could still be applied. However, new challenges for security arise to the research community.

An example of a fully distributed recommender system is a distributed filesystem where users publish their resources (movies, music, personal data, etc.) and ask for recommendations to their friends about interesting resources. This is the scenario that we will use for the rest of this document.

Secure recommender system. Publishing personal resources in a distributed filesystem and asking for recommendations are two very sensitive processes from the point of view of security and privacy. Many people won't like that their personal resources are stored in uncontrolled nodes all over the Internet without protection. In addition, a recommendation is a very personal process that contains lots of private data. The final recommendation depends on the users' likes and dislikes, and the actor in charge of making a recommendation need these likes and dislikes to give a useful piece of advice. If customers are afraid of publishing their likes, then recommendations are going to be less accurate. Then, protecting the user's privacy is not only interesting for the user, but also for the recommender since it enhances the recommendation results.

Furthermore, we will show that other actors of the recommendation process need additional protection. The provider of a recommendation, for example, exposes his own opinion of the resource that he is recommending. Moreover, copyright infringements of the participants of the system must be taken into account.

1.1. Definition of a recommender systems

The introduction and first definition of collaborative filtering was proposed in [50] twenty years ago, in the shape of a community-based system to annotate mails as interesting or not. Since the publication of this seminal work, the terminology and definitions of recommendation or recommender systems have been extended to cover additional mechanisms and more complex environments. Different definitions found in the literature include [50, 108, 68]. However, from this point forward, we will consider a recommender system as defined next.

A recommender system is an automatic system that, given a user u and a set of items S some of them unknown by u, returns a personalized subset including the most interesting items of S for the user u.

We stress some key aspects of this definition:

- It is an *automatic* system
- It outputs resources that *the user wasn't aware* of their existence
- It provides a *personalized* output for each user

These three main features can be considered mandatory in every recommender system. However, our view of a recommender system includes two additional characteristics:

- It should take care of the *whole process*, including not only recommendations but also the final delivery of the resources.
- It should be *secure*.

Automatic system. Recommender systems have been always used. We refer to them as *experts*, and we still come to them in some fields such as medicine, financial support or personal training. Automatic recommender systems come as a substitute for experts, not only because using automatic systems is usually much cheaper than hiring experts, but also because we really believe that an automatic recommender system can outperform the advice of an expert. Maybe, we are not prepared yet to trust on an automatic recommender system in cases such as health services, but recommender systems provide definitely better support than experts in fields such as e-commerce or financial investments.

Discover previously unknown documents. During a long time, the discovery of interesting information in the Internet has been a painful process. A very specialized machine with an incredible amount of memory and bandwidth crawled the net for identifying, collecting and parsing as many pages as possible, and inserted these data into a huge database. Meanwhile, users sent to the centralized database a list of keywords and retrieved a set of pages containing these words. Altavista, Yahoo, Google, Bing and many others worked and still work this way. These web search engines work quite well as long as users know in advance the contents of the resource that they are looking for. However, the list of words must be so specific and well-defined that it is very difficult to find something users are oblivious about. While search engines assist you in the process of looking for the things that you know, discovery engines help you in finding the rest. We consider recommender systems as discovery engines that find interesting and previously unknown items.

Personalized recommendations. When we play the role of buyers and enter the market, and this is something that we do on an everyday basis, we make lots of decisions about the products that we are looking for: which is the best movie currently at the theater, the most suitable computer to buy or the most interesting book to read. Gathering all the information to make a well-grounded decision is a very time consuming process. Recommender systems appeared to assist the user in quickly making the right decision and saving time and maybe money. Many online shops decided that recommender system were an additional service to offer to their clients. For example, this is the case of Amazon or Netflix. Other enterprises made of appropriate recommendations their core business. This is the case of FilmAffinity or LibraryThing.

In order to achieve this, a recommender system creates a model of users that captures their needs or preferences, defines a mechanism to describe resources and includes an objective way to calculate how interesting these resources are for the users of the system. In order to provide a satisfactory buying experience, all these processes should be as unobtrusive as possible.

A complete process. Recommender systems are extensively used in electronic commerce. From the point of view of a user, the output of a commercial exchange is an item, a product or a service. However, many academic recommender systems do not consider the final delivery of the real resource part of the recommendation process, and returns only a reference to the recommended item. Commercial recommender systems, on the other hand, understand that there is no point in discovering an item that cannot be accessed, bought or downloaded. The result of the process can be used as feedback for improving future recommendations. Giving a way to access the recommended item is even more important in distributed networks, where any node of the network may recommend any resource, and there must be a mechanism for the user to trace the item back.

Thus, we establish that the recommendation process is not over until the recommended object is provided to the user.

Security of the users. The process of receiving a useful recommendation begins with the creation of a view of the user that contains their likes and dislikes. Thus, user's profiles include sensitive information which captures the personal description of a particular user. Protecting the users' privacy is not only a necessity for users, since it can improve the result of the recommendation process. Indeed, if users are not afraid of declaring their likes and dislikes, the recommendations that they get from the system will be more accurate. Protecting the user's privacy is not the only security service to provide in recommender systems. In a distributed environment, other actors of the system may need additional protection. The providers of a recommendation, for example, expose their own opinion of the resource that they are recommending. Thus, recommenders should be protected in the same way than users. Furthermore, recommenders and other participants that assist in the recommendation process may be, even unknowingly, committing a copyright infringement. The risk of being prosecuted may affect the quality of the output of the system, for example, preventing the recommendation of a certain movie even if the recommender thinks that it is the most suitable for the user. This is a new kind of legal attack, and providing protection for this attack may improve the quality of the recommendations.

1.2. Steps of a recommendation

Last section introduced the definition and main features of a recommender system. In this section, we formalize the steps that a recommender system takes for providing a recommendation. We aim for a general description of a recommender system, trying to fit many different types within the same structure. Thus, actual systems may reduce some of these steps to a simple process while others perform complex tasks inside them.

Document collection and profiling. During this step, the recommender system collects and identifies the items that it is going to offer to the users. For example, in a centralized online shop, the merchant inserts the items in the local database and adds a description to them. If the recommendation is based on the properties of the items, this step includes the creation of a profile that captures the defining characteristics of the item. As we will see in chapter 2, some recommender systems do not need to assign a profile to the resources.

User profiling. During this step, the user enters the system. This is the sign-on process of an online shop, or the first run of a P2P client in a distributed recommender system. In this moment, a user profile is created and assigned to the user. This profile could be controlled by the user, for example, if it is based on the answers of a test, or controlled by an external observer. This is the case of profiles that involve the study of the buying habits of the users. Therefore, the information that is included in the user profile is highly sensitive, and the system must provide mechanisms to protect and secure this profile.

Recommender selection. The recommender system may have many actors that are able to answer to a recommendation request. For example, some recommenders may be specialized only on some product categories, or it is better to select a recommender that shares interests with the user. During this step, the system selects those recommenders that are more suitable to answer the query of the user. For example, the number of users in movie recommendation systems may be of millions, and the number of available movies is several thousands. In order to manage this amount of information, an initial classification of users that rate similar the same movies takes place. In a social network, participants often select an initial set of “friends” or “similar people” that can be used to make recommendations.

Query the system. During this step, users send a query to the system that includes their current interests. The complexity of the process of querying the system varies with the different recommender types. For example, this is a very simple process in an online shop, since it is reduced to entering a text in the home page of the shop. In distributed systems, on the other hand, this step involves routing the query to the selected recommenders and it may be a complex task. As in the case of users’ profiles, the query of a user to the system includes sensitive information that must be protected.

Recommendation process. The selected recommenders search their internal resource databases to select those resources that are more suitable to answer the query of a user. Then, the recommenders return a set of links to the resources that they believe that are interesting to the user.

At this point, we find useful to imagine a recommender system as a matrix where rows are users and columns resources, as figure 1.2 shows. An element of the matrix r_{ij} is the rate that a user i gave to a resource j. This matrix is scarcely populated and most of the elements are empty, since it is usually impossible for users to evaluate a significant subset of the available resources. The goal of the recommender system is making a good guess of the rate that a user would give to a resource that is not yet evaluated. Given these guesses, the system decides whether a resource interests the user or not with an algorithm that is often as simple as “the document is interesting if its calculated rate is higher than a threshold λ”. The mechanisms that are used to populate the elements of the matrix, the input that the recommender needs for guessing rates and the actual location of the matrix in the system are the main differences between the different implementations of real recommender systems.

Accessing the recommended resources. During the final phase of the system, users access the recommended items. In online shopping, users buy and receive the selected products. In a streaming multimedia service, users access and watch the recommended movies. The final output of the process may be useful to enhance future recommendations. This is the case of user profiles that are based on buying habits. From the security point of view, an access to a resource implies that a recommendation was correct, and this is a security leakage. Even if the other steps of the process are conveniently protected, an attacker may learn something about a user’s profile by means of inspecting only the resources that the user downloads. A system that aims to protect the user’s privacy must consider the final access to the item as part of the recommendation process, and provide protection to this access as much as it protects queries and/or recommendations.

	Doc 1	Doc 2	Doc 3	Doc 4
User A	0.9	-	-	-
User B	-	0.5	0.8	0.5
User C	0.2	-	0.8	-
User D	0.1	0.8	-	-

Figure 1.1.: View of recommender systems as a matrix of users and interests

1.3. Actors and roles in a recommendation system

Figure 1.2 shows the main roles and interactions of a recommender system. As recommender system and e-commerce are heavily interrelated, we will use a similar terminology to describe many of the actors of a recommender system.

Documents are the resources that costumers are looking for. They may be real objects, such as books, videos, or electrical appliances, accessible services, or a composition of business processes that achieves some desired goal. Documents may be modeled using a **document profile**. During this thesis, we establish that documents can be streamed or downloaded from the Internet.

Customer is the role of the actor that looks for recommendations of documents. Customers are modeled with a **customer profile** that captures their likes and dislikes and/or issue **queries** that contain a description of the document that they are currently interested in. Customers may be aware of the existence of their profile or not. In the first case, costumers can have access to their profile (for example, to modify or obscure it) or not.

Merchant is the role that an individual or organization plays when he offers a document to the community. If document profiles exist, the merchant is in charge of providing them. Additionally, there may be a **merchant profile** under the same conditions than costumer profiles. The documents of the merchant are stored in a document repository.

Profiler is the role of the system played by the entities that match users' queries and suitable recommenders. A profiler selects the recommenders that are more likely to answer the query of the user.

Recommender is the role of the system played by the entities that match customer profiles and document profiles. Recommenders output a link to an interesting document that can be used to locate the document in the document repository.

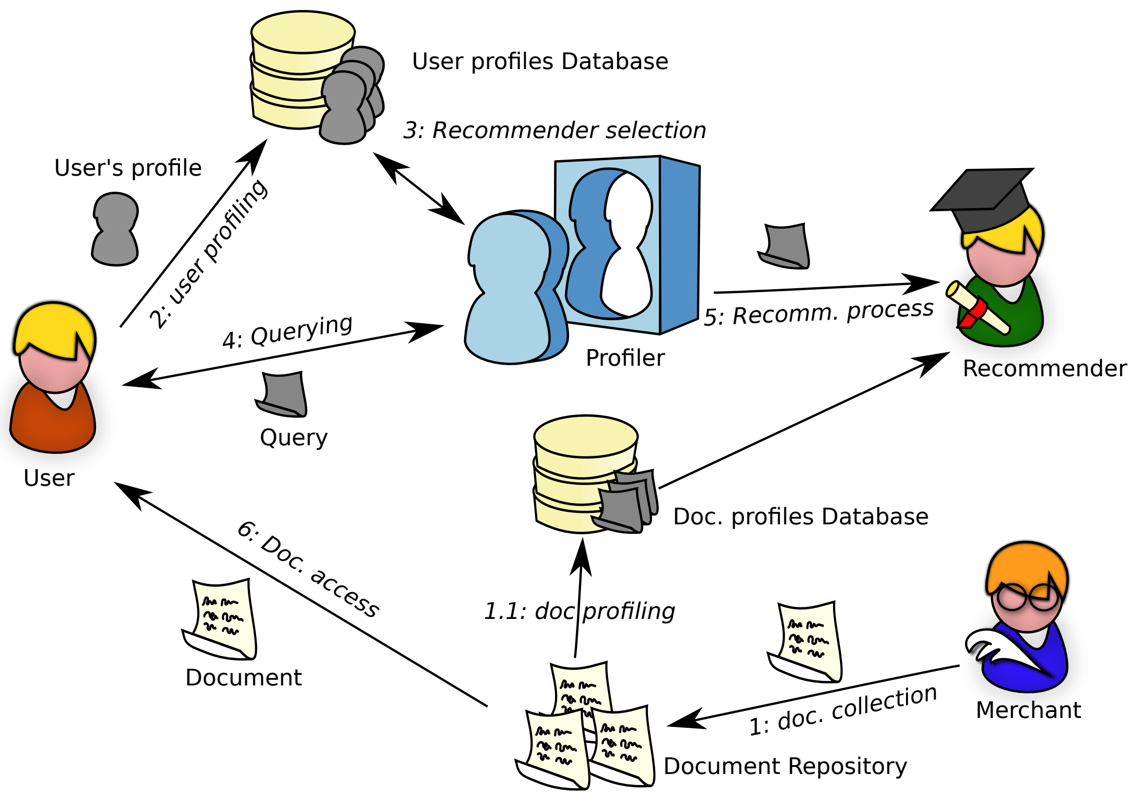


Figure 1.2.: Actors and interactions of a recommender system

Document profile database is a database of the profiles of the documents that the merchants share. It may be distributed.

User profile database is the actor that stores customers and merchant profiles, if they exist.

Document repository is the actor that stores documents for the merchants. Given a link to a document from the recommender, this actor returns the document that the costumers are interested in.

Some actors of the system may play different roles at different moments of the recommendation process. For example, in a distributed recommender system, it is common that users play both the role of costumers and merchants. From this point forward, we will use the term **users** when the exact role of customers or merchants they are playing is unimportant. In most centralized recommender systems, the costumer profile resides in the central node that plays the roles of user profile database, document profile database and recommender, at least.

2. A taxonomy of recommender systems

Previous works in the literature created categories of recommender systems to extensively classify all of them [9, 68, 59]. Every classification is necessarily a simplification, since real recommender systems often share elements of several categories. In any case, such categories are useful to understand the current trends in the automatic recommender systems field, and some of the decisions that their designers took.

In this thesis, we will classify a recommender system based on three different axes:

- According to the source of data for recommenders. The recommender system may use the past behavior of the customer as the main hint to select documents. Or it might analyze the structure of the document to decide which one is more interesting to the customer.
- According to the structure of the system. This way, recommender systems can be centralized, decentralized or organized as a social P2P network.
- According to the security that the system offers to its actors. This axis captures whether or not the system protects the privacy of the users, the confidentiality of the documents, the issuer of a recommendation or the distributor of the final document.

2.1. According to the source of data for recommenders

According to the source of the data that is used to create a recommendation, the systems are divided in (i) collaborative filtering (recommendations are based on the evaluations of our friends), (ii) content based filtering (recommendations are calculated by direct inspection of the characteristics of the documents) and (iii) knowledge based recommendation (performed by an expert that surveys the interest of the customer in a personal way). Most real systems do not use a pure approach, and prefer (iv) a hybrid algorithm to decide recommendations.

2.1.1. Collaborative filtering

In collaborative filtering, users help each other to detect interesting documents. A recommendation that uses collaborative filtering only detects the *annotations* that other users made of the available documents, and therefore a recommendation does not depend on the *content* of the documents. In this context, an annotation is the evaluation or rate that a user assigned to the document in the past. In collaborative filtering recommender systems, the recommender guesses the annotation that a specific customer will assign to a document by means of aggregating the annotations provided by other users, weighted with the similarity between users. Often, to save resources and computing time, the recommender does not use the whole set of available users but only a subset of it. This subset includes only the *friends* of the original users.

Collaborative filtering is based on these assumptions:

- Two users that showed common interests in the past will show common interests in the future. As a consequence of this property, annotations that other users made are useful to decide whether or not a document is interesting for a customer.
- No additional information apart from annotations is necessary. Especially, information about the item (genre, author, etc.) is considered redundant and the recommender does not use these data to issue a recommendation.

Collaborative filtering is one of the most successful type of recommender system in the literature. Many different recommender systems use collaborative filtering, since analyzing documents is a difficult task for the current state-of-the-art artificial intelligence, and categorizing them is still a task where humans outperform machines. In this case, observing the behavior of humans and not analyzing documents seems a reasonable and easy way to provide useful recommendations.

Unfortunately for the recommender, the process of collecting annotations for every available document may be hard and slow. This collection can use explicit or implicit methods. Some examples of explicit data collection are: (i) asking the customer to rate an item on a sliding scale, (ii) asking a user to rank a collection of items from favorite to least favorite, (iii) showing two items to a user and asking him to pick the better one, or (iv) asking a user to create a list of items that he likes. Examples of implicit data collection include the following: (i) observing the items that a user views in an online store, (ii) analyzing item/user viewing times, (iii) keeping a record of the items that a user purchased online, (iv) obtaining a list of items that a user has listened to, and (v) analyzing the user's social network and discovering similar users.

One of the most used algorithms in collaborative filtering is the k-nearest neighborhood approach. In a social network, a particular neighborhood of users with similar tastes or interests can be created by calculating the Pearson's correlation between the users. This way, friends of the original user are identified and linked together, creating a neighborhood of users that share interests.

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}} \tag{2.1}$$

$$pred(a,p) = \bar{r}_a + \frac{\sum_{b \in N} sim(a,b)(r_{b,p} - \bar{r}_b)}{\sum_{b \in N} sim(a,b)} \tag{2.2}$$

Being, $sim(a,b)$ the similarity of two users a, b (or *Pearson's correlation coefficient*), $\bar{r}_a$ the average rating of user a, and $r_{a,p}$ the rating of user a of document p, and $pred(a,p)$ the prediction of the rating that user a would give to document p. [58, 68, 13] compares this metric with the cosine similarity and Bayesian networks. The current opinion between the experts seem biased to consider the cosine metric (to be introduced later in this document) as superior to this similarity.

By means of collecting the preference matrix of top-N nearest neighbors of the particular user (weighted by the similarity of the users), the evaluation that a user will make about an unknown document can be predicted. The approach of creating a neighborhood of friends has an additional advantage: the system does not have to evaluate the global set of documents, since evaluating the documents that are known by the neighborhood is usually enough.

This model shows other drawbacks that are intrinsic to the collaborative filtering concept, and they are difficult to cope with.

- The model suffers of the sparsity problem of the evaluation matrix. Many documents are annotated by few users, and only some of them are really popular. The creation of useful neighborhoods without sharing some evaluations of documents is not possible in a pure collaborative filtering approach.
- Despite the last point, [22] found that an agreement on popular documents is not as important as an agreement on non-popular documents. In their work, authors reduced the relative importance of the agreement on universally likable items using a variance weighted factor.
- [58, 59] showed that users that rated few items are a poor input for recommenders. They introduced the necessity of a significance weighting. This is a problem similar to the "gray sheep problem", which studies how to identify and manage those users that are not consistent with their annotations. Unfortunately, most users if not all of them follow a "gray sheep" behavior in some of their annotations.
- Most collaborative filtering uses the k-neighbors approach. However, if the threshold to accept a new friend in the neighborhood is very restrictive, a reduced coverage of the available data is produced and data sparsity appears [58]. [59] showed that "in most real-word situations, a neighborhood of 20 to 50 neighbors seems reasonable".
- Implicit methods to collect annotations such as analyzing the browsing history maybe heavily intrusive and unacceptable by the user. If the user takes some protection techniques such as anonymous/private browsing and cannot be identified, it might be not possible to give him personalized recommendations.
- The new item problem: when a new document is introduced in the system, no user has inserted any annotation and won't appear in any recommendation. Different methods to publish new items in the system must be included in collaborative filtering systems. Similarly, new users without any prior history or profile can get

recommendations, especially of generally liked documents and hits. The accuracy of the recommendations increases with the gradual definition of the user's profile.

Next, we study some examples that use collaborative filtering as the main input for their recommenders.

Amazon is a paradigmatic example of collaborative filtering is the recommender system [12]. A customer with a long history that enters Amazon finds recommendations in three different categories: "customers that bought X also bought Y", "customers that visited X also visited Y" and "X% of customers that visited this page bought the product". Some additional parameters that describe the item are used in some extend, such as recommendations of books by the same author, other books of the same category, new books that entered an interesting category.

The recommender system of Amazon is intentionally simple and focused mainly on implicit information of the user collected from his behavior and not his description. Steven Johnson, during an interview in Amazon.de, explains these points: "The software doesn't know what it's like to read a book, or what you feel like when you read a particular book. All it knows is that people who bought this book also bought these other ones; or that people who rated these books highly also rated these books highly, etc. Out of that elemental data something more nuanced can emerge–if you set up the system correctly, and give it enough data."

Based on the success of Amazon on collaborative filtering, there are other recommender systems that share the same ideas.

MovieLens [55] is a recommender system for movies that started as an academic sandbox project. When a user joins the system, he is presented a list of random movies to rate from 1 to 10. Then, similar users are identified according to these rates, and an estimation of interesting movies is performed using collaborative filtering. **FilmAffinity** [7] is a commercial system that offers a service that is very similar to MovieLens, but the initial list of movies to rank is not random. FilmAffinity presents in the initial list of movies to rate well-known and highly controversial movies. In this case, the initial profile of the user is more extreme than MovieLens.

StumbleUpon [121] is a web page recommender that uses pure collaborative filtering based on user profiles. After joining, users create a profile of their interest by manually stating the set of categories and subcategories that they are interested in. In total, there are about 500 different categories and subcategories. Furthermore, the user install a simple bar in his browser with three main buttons: "like", "dislike", and "stumble!". While browsing, the user can express if he liked/disliked a page, or request a random page that is picked up according to the pages that other similar users liked.

Some other collaborative filtering recommender systems are marketed as solutions to e-business and not linked to a specific shop or site.

Strands [120] is a plug-in to existing e-shops that analyzes the behavior of the user inside the shop, and presents product recommendations based on the items that other users with similar behavior visited.

AggregateKnowledge [11] is based upon the idea that "the tastes of your friends poorly reflect yours". It focuses on the context of the visit (traffic source, semantics, etc.), the behavior of the visitors (page views, clicks, etc.) and analysis of the items that they present.

2.1.2. Content-based recommendation

A recommendation is a way to cope with information overload, by means of categorizing documents in adjacent sets of interesting and not interesting documents. In we explore recommender systems from the point of view of the document contents, they are strongly related to the information retrieval and information filtering fields. In these fields, a document is classified as interesting/not interesting according to its contents and/or metadata, and not the subjective opinion of external users about the document [96, 124, 125, 86].

A content-based recommender system analyzes the contents of the available documents to provide recommendations to the customers. This approach to recommender systems shows these main ideas:

- Recommenders must classify documents in two sets, relevant and not relevant documents, for all customers [96].
- The classification of the documents must take into account the profiles of these documents. The opinion that users have on some documents is usually not relevant.
- The inputs for the recommender are the document profiles and (optionally) the customer profile to test the suitability of the classification to a specific user.
- As a consequence, the recommender can explain the reasons for a specific recommendation.
- Profilers must analyze documents, collect all significant information and take a decision about their relevancy. Often, these techniques impose some internal structure to the documents, or the existence of specific metadata prior to their insertion into the system. The creation of this data structure is usually handed by the merchant.
- The source of the information to create a recommendation is the document and not the user profile or preferences [45]. Hence, documents can be evaluated and categorized as they are published. The content-based approach does not suffer from the new document problems, as the collaborative filtering methods do. However, they suffer from the cold start problem: a new user must build his profile before getting any recommendation [144, 66]
- Sometimes, classification is not objective and based on human parameters. For example it is difficult to capture the “ease of use” of a computer science book. This kind of data is better managed by collaborative filtering methods

- Content-based recommendation shows a problem of over-specialization. For example, if a user never showed interest on Greek cuisine, he will never get a recommendation about the best Greek restaurant in the town.
- Content-based recommendation must cope with synonymia and similarity. Two documents that are extremely similar may be actually the same document and the system must identify and manage these situations.

In content-based recommender systems and during the insertion of a document into the system, the document profiler must model the document using a profile. Techniques to classify these profiles include Bayesian classifiers, neuronal networks and cluster analysis. One of the most successful representations of a document in the Information Retrieval field is the use of vectors [85, 124]. The components of these vectors are the frequency of appearance of certain common terms in the document under analysis. Given a predefined set of terms, a document's profile can be constructed analyzing the weight of each term inside the document.

$$tf_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}} \tag{2.3}$$

$$idf_t = \log(\frac{N}{df_t}) \tag{2.4}$$

$$w(t,d) = tf_{t,d} \times idf_t \tag{2.5}$$

Being $n_{t,d}$ the number of appearances of the term t in the document d, $tf_{t,d}$ the frequency of a term in a document, N the number of documents, df_t the number of documents that contain the term t, idf_t the inverse document frequency and $w(t,d)$ the weight of a term in a document. The document's profile is the set of the weights of the analyzed terms. This way of constructing a document profile is called tf–idf weight (term frequency–inverse document frequency).

This same idea can be generalized to include taxonomies or categories of terms. Indeed, it is possible to convert a vector of terms (*bag-of-words*) into a vector of taxonomies or categories (*bag-of-concepts*). The creation of a taxonomy that classifies documents falls beyond the scope of this thesis. In interested, the reader may refer to recent studies in the Information Retrieval and Artificial Intelligence fields [85, 125, 86, 129].

One of the main metrics to calculate the similarity of two documents is the cosine metric of Equation 2.6. In [147] we find simulations using the cosine metric in a P2P network, and it is proved that it may enhance search results [106, 85, 79]. We will study this metric in chapter 5, while modeling our social space.

$$sim(\bar{p}_1, \bar{p}_2) = \frac{\bar{p}_1 \cdot \bar{p}_2}{|\bar{p}_1||\bar{p}_2|} = \frac{\sum p_{1_i} p_{2_i}}{\sqrt{\sum p_{1_i}^2 \sum p_{2_i}^2}} \tag{2.6}$$

PANDORA [94] is one of the most successful examples of content-based recommenders. Pandora is a music recommender radio that analyzes music files to detect the inner structure of the music such as patterns, rhythm, vocals, lyrics, etc. Then, the profiler builds a database of songs and proposes to its users a list of the songs that are more suitable to their tastes. Pandora uses the Music Genome project [48] to analyze up to 400 different attributes and to create a classification of the song. Any song is represented by a vector that contains approximately 400 attributes or so-called *genes* and each gene corresponds to a characteristic of the music. For example, the rhythm of the music, the type of instrumentation, the recording style, influences, individual instruments, etc. The creators of the Music Gene project reported that rock and pop songs have 150 genes, rap songs have 350, and jazz songs have approximately 400. The system depends on a sufficient number of genes to render useful results. Each gene is assigned a number from 1 to 5, in half-integer increments. The music Genome project plays the recommender role of this system. It calculates offline the genes of every song and the distance between them. The customer selects a song that he likes, and the recommender points to similar songs that he may like. Finally, Pandora provides a streaming service to listen to the recommended songs.

Internet Movie Database [67] is another example of a content-based recommender system. When a customer visits the description of a movie in the web, the recommender shows a “if you liked this movie, our system recommenders...” list of movies. This list is constructed using movie ratings from other customers, and analyzes the genre, title and keywords of the movies [45]. Customers gain access to this list of recommendations even if they never visited IMDB before, and hence without providing any personal information.

Webwatcher [70] and [97] are other representative cases of content-based recommender systems. They were early web crawlers that inspected web links to categorize them and present the results to the customers.

Due to the limitations of this type of system, currently cooperative filtering techniques seem to be favored by both the commercial and academic recommender systems.

2.1.3. Knowledge based recommendation

Closely related to the expert systems, knowledge-based recommender systems provide a set of interesting documents based on the characteristics of a product and the current interest of the customer. They are used when a list of products that fulfill a set of characteristics is not enough, since it lacks personalization, or the costumer is not sure about the list of characteristics that he needs. This is the kind of recommendation that we get from a salesperson in a department store. We look for a dishwasher, and by no means are we experts on this kind of appliance. Then, we ask the salesclerk for a recommendation and he will ask questions about the capacity that we need, or the energy consumption that we are able to afford, the kind of dishes that we own and so on.

Knowledge based recommendations have these main points:

- Usually, items are classified in a very specific personal way and for a “one time buyer”. The system cannot trust on the past history of the customer to decide whether or not an item is interesting for him. The survey to identify the interesting documents must be taken during each interaction with the system.

- It is difficult to capture the “relative importance of characteristics”, and depends on the skills of the seller or the designer of the survey.
- This kind of system needs a complex interaction between the customer and the recommendation system
- Misbehavior of the seller apart, the final recommendation is highly accurate and personalized to the exact needs of the customer.

StyleHop [2] predicts fashion trends incorporating consumer preferences, risk and other data. Its service MarginFeeder collects data from specifically targeted, trend conscious consumers. StyleHop came to an end on 2010, but similar services such as **ModCloth** still exist for more specific fashion markets [72].

Some other recommender systems use social networking to create recommendations. This is the case of review aggregators and expert ratings, such as **TrustPilot**, **Ciao** or **Yelp** [130, 115, 148]. These systems have not any automatic recommender, and they allow customers to comment and read comments on specific products before buying. The final decision is made by the customer, based on the opinions of other users of the web about the same product. Sometimes, these systems include a content-based recommendation by means of lists of “related and similar products”.

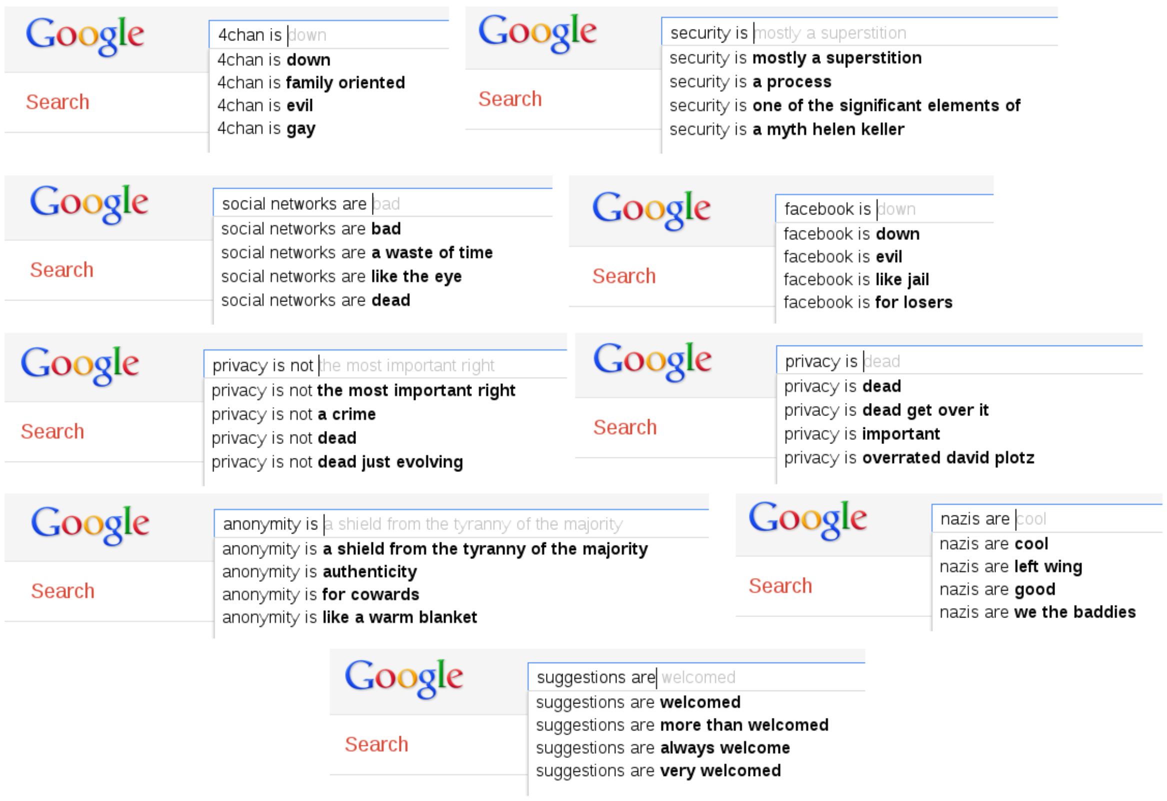
A especial form of expert recommenders are darknet networks that are organized as “a network of friends”, such as **RetroShare** [107], **Gnunet** [74], **i2p** or **anoNet**. In these networks, users that trust each other link in a darknet and share the list of documents that they own [109]. As far as we know, these networks do not include any form of automatic recommender, but the fact that nodes are connected only to *friends* shows that they share something in common. Therefore, the list of documents that our friends own may be used to get a basic kind of expert recommendation.

The knowledge based recommender system is not of the interest of this thesis. Making automatic recommendations in a social network, as we intend in this thesis, seems to be against the specific opinion of a human, which is the basis for knowledge based recommendations.

2.1.4. Hybrid approaches

The mechanisms that were introduced in the last sections are rarely used in isolation. For example, Amazon adds to its collaborative filtering mechanisms additional criteria such as others books by the same author, or other books in the same category. Most recommender systems share several properties from different categories. This way, most recommender systems use hybrid approaches to create the list of recommended items. In this section, we analyze Google, which is a clear example of a hybrid recommender system.

Google [52] is a paradigmatic example of a hybrid recommender system. When customers search in Google, they enter an expression containing keywords and sentences. Then, Google shows a list of pages that contain the list of keywords. Google also takes into account synonyms, miss-spelling and usually related keywords to tweak the list of keywords, including suggestions such as ”did you mean...?”. An example of these suggestions are shown on figure 2.1. The keyword list is the main filtering criteria for links, and corresponds to a content-based recommender system. However, this list is ordered based on the PageRank algorithm. In this algorithm, pages are ranked according to the number of external sites that that link back to the page, and the rank of these external sites. Additionally, Google uses some context information about the user to modify the order of the links. For example, the final list is heavily based on the location of the customer and his previous web history. The parameters to order the list of links are a case of collaborative filtering recommender.



4chan is down
4chan is **down**
4chan is **family oriented**
4chan is **evil**
4chan is **gay**

security is mostly a superstition
security is **mostly a superstition**
security is **a process**
security is **one of the significant elements of**
security is **a myth helen keller**

social networks are bad
social networks are **bad**
social networks are **a waste of time**
social networks are **like the eye**
social networks are **dead**

facebook is down
facebook is **down**
facebook is **evil**
facebook is **like jail**
facebook is **for losers**

privacy is not the most important right
privacy is not **the most important right**
privacy is not **a crime**
privacy is not **dead**
privacy is not **dead just evolving**

privacy is dead
privacy is **dead**
privacy is **dead get over it**
privacy is **important**
privacy is **overrated david plotz**

anonymity is a shield from the tyranny of the majority
anonymity is **a shield from the tyranny of the majority**
anonymity is **authenticity**
anonymity is **for cowards**
anonymity is **like a warm blanket**

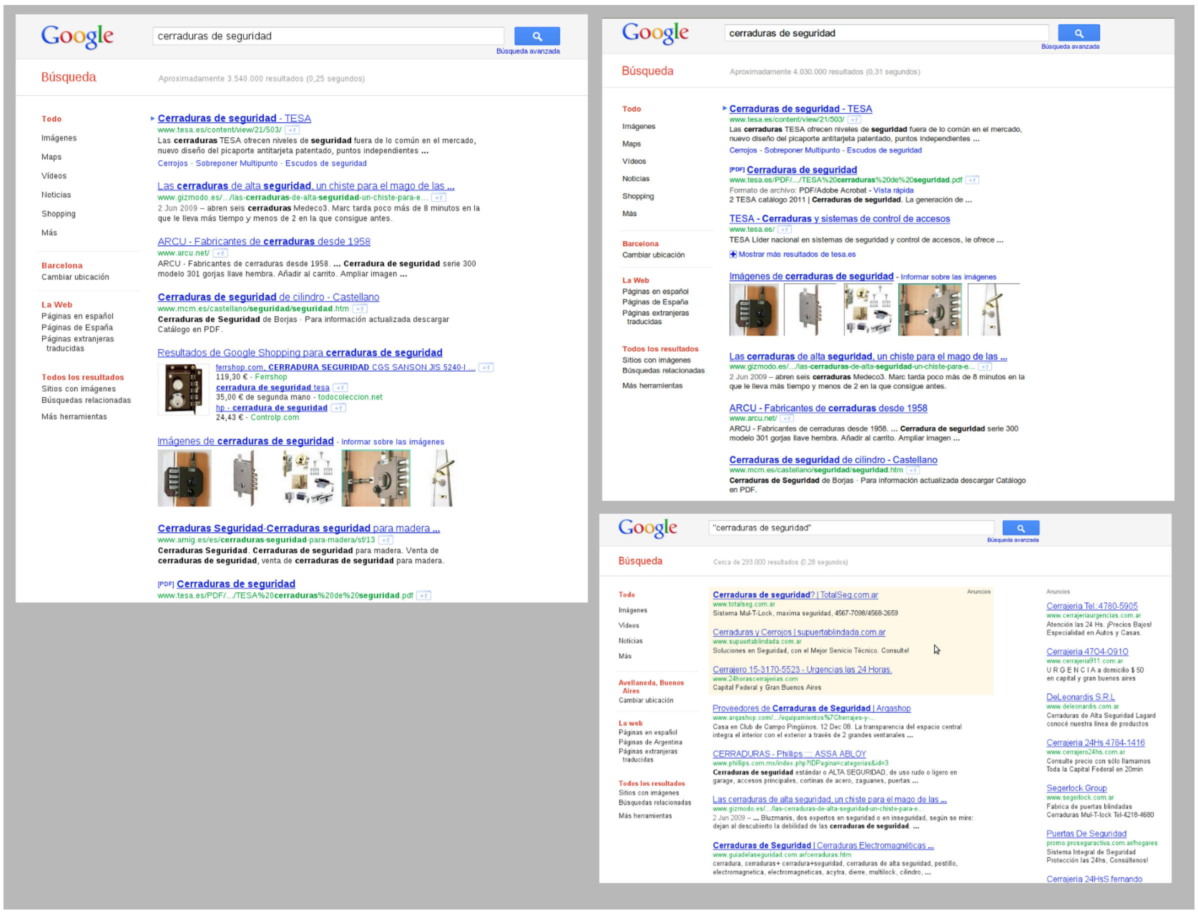
nazis are cool
nazis are **cool**
nazis are **left wing**
nazis are **good**
nazis are **we the baddies**

suggestions are welcomed
suggestions are **welcomed**
suggestions are **more than welcomed**
suggestions are **always welcome**
suggestions are **very welcomed**

Figure 2.1.: Several suggestions from Google Suggests

Eli Pariser [95] alerted that the smart algorithms that are used to recommend pages in Google and other recommenders are deciding which content is relevant to our interests and leaving out, till disappearance, any other content. Eli Pariser introduced the concept of the “filter bubble”, which is created when automatic systems recommended always similar items and only show documents that are of our interests. This way, it is increasingly harder to find new content or different points of view that are located outside the automatic filter bubble.

We performed a casual experiment to test these ideas. The sentence ”security locks” was searched by users that (i) were never interested on locks, (ii) were actively looking for locks in the past weeks, and (iii) were familiar with locks but not searching them specifically. User (i) was located in Argentina, while users (ii) and (iii) were in Spain. Results are shown in Figure 2.2. User (i) received generic information about locks. User (ii) found that nine out of ten links of his first page were to the company TESA, or reviews for TESA's products. Any other lock from a different maker was not presented in his first page. User (iii) received a page similar to the one received by user (ii), but without the strong bias towards the products by TESA. User (ii) acknowledged that without comparing his results with the results obtained by other users, he would never consider a lock not made by TESA.



Aproximadamente 3.540.000 resultados (0,25 segundos)

Cerraduras de seguridad - TESA
www.tesa.es/content/view/21/503/
Las cerraduras TESA ofrecen niveles de seguridad fuera de lo común en el mercado, nuevo diseño del picaporte antitarjeta patentado, puntos independientes ...
Cerrojos · Sobreponer Multipunto · Escudos de seguridad

Las cerraduras de alta seguridad, un chiste para el mago de las ...
www.gizmodo.es/.../las-cerraduras-de-alta-seguridad-un-chiste-para-e...
2 Jun 2009 – abren seis cerraduras Medeco3. Marc tarda poco más de 8 minutos en la que le lleva más tiempo y menos de 2 en la que consigue antes.

ARCU - Fabricantes de cerraduras desde 1958
www.arcu.net/
ARCU - Fabricantes de cerraduras desde 1958. ... Cerradura de seguridad serie 300 modelo 301 gorjas llave hembra. Añadir al carrito. Ampliar imagen ...

Cerraduras de seguridad de cilindro - Castellano
www.mcm.es/castellano/seguridad/seguridad.htm
Cerraduras de Seguridad de Borjas · Para información actualizada descargar Catálogo en PDF.

Resultados de Google Shopping para cerraduras de seguridad
ferrshop.com, CERRADURA SEGURIDAD CGS SANSON JIS 5240-I ...
119,30 € - Ferrshop
cerradura de seguridad tesa
35,00 € de segunda mano - todocoleccion.net
hp - cerradura de seguridad
24,43 € - Controlp.com

Imágenes de cerraduras de seguridad - Informar sobre las imágenes

Cerraduras Seguridad-Cerraduras seguridad para madera ...
www.amig.es/es/cerraduras-seguridad-para-madera/sf/13
Cerraduras Seguridad. Cerraduras de seguridad para madera. Venta de cerraduras de seguridad, venta de cerraduras de seguridad para madera.

[PDF] Cerraduras de seguridad
www.tesa.es/PDF/.../TESA%20cerraduras%20de%20seguridad.pdf

Aproximadamente 4.030.000 resultados (0,31 segundos)

Cerraduras de seguridad - TESA
www.tesa.es/content/view/21/503/
Las cerraduras TESA ofrecen niveles de seguridad fuera de lo común en el mercado, nuevo diseño del picaporte antitarjeta patentado, puntos independientes ...
Cerrojos - Sobreponer Multipunto - Escudos de seguridad

[PDF] Cerraduras de seguridad
www.tesa.es/PDF/.../TESA%20cerraduras%20de%20seguridad.pdf
Formato de archivo: PDF/Adobe Acrobat - Vista rápida
2 TESA catálogo 2011 | Cerraduras de seguridad. La generación de ...

TESA - Cerraduras y sistemas de control de accesos
www.tesa.es/
TESA Líder nacional en sistemas de seguridad y control de accesos, le ofrece ...
Mostrar más resultados de tesa.es

Imágenes de cerraduras de seguridad - Informar sobre las imágenes

Las cerraduras de alta seguridad, un chiste para el mago de las ...
www.gizmodo.es/.../las-cerraduras-de-alta-seguridad-un-chiste-para-e...
2 Jun 2009 – abren seis cerraduras Medeco3. Marc tarda poco más de 8 minutos en la que le lleva más tiempo y menos de 2 en la que consigue antes.

ARCU - Fabricantes de cerraduras desde 1958
www.arcu.net/
ARCU - Fabricantes de cerraduras desde 1958. ... Cerradura de seguridad serie 300 modelo 301 gorjas llave hembra. Añadir al carrito. Ampliar imagen ...

Cerraduras de seguridad de cilindro - Castellano
www.mcm.es/castellano/seguridad/seguridad.htm
Cerraduras de Seguridad de Borjas · Para información actualizada descargar Catálogo en PDF.

Cerca de 293.000 resultados (0,28 segundos)

Anuncios
Cerraduras de seguridad? | TotalSeg.com.ar
www.totalseg.com.ar
Sistema Mul-T-Lock, maxima seguridad, 4567-7098/4568-2659

Cerraduras y Cerrojos | supuertablindada.com.ar
www.supuertablindada.com.ar
Soluciones en Seguridad, con el Mejor Servicio Técnico. Consulte!

Cerrajero 15-3170-5523 - Urgencias las 24 Horas.
www.24horascerrajerias.com
Capital Federal y Gran Buenos Aires

Proveedores de Cerraduras de Seguridad | Arqashop
www.arqashop.com/.../equipamientos%7Cherrajes-y-...
Casa en Club de Campo Pingüinos. 12 Dec 08. La transparencia del espacio central integra el interior con el exterior a través de 2 grandes ventanales ...

CERRADURAS - Phillips :::: ASSA ABLOY
www.phillips.com.mx/index.php?IDPagina=categorias&id=3
Cerraduras de seguridad estándar o ALTA SEGURIDAD, de uso rudo o ligero en garage, accesos principales, cortinas de acero, zaguanes, puertas ...

Las cerraduras de alta seguridad, un chiste para el mago de las ...
www.gizmodo.es/.../las-cerraduras-de-alta-seguridad-un-chiste-para-e...
2 Jun 2009 – ... Bluzmanis, dos expertos en seguridad o en inseguridad, según se mire: dejan al descubierto la debilidad de las cerraduras de seguridad. ...

Cerraduras de Seguridad | Cerraduras Electromagnéticas ...
www.guiadelaseguridad.com.ar/cerraduras.htm
cerradura, cerraduras+ cerradura+seguridad, cerraduras de alta seguridad, pestillo, electromagnetica, electromagneticas, acytra, dierre, multilock, cilindro, ...

Anuncios
Cerrajeria Tel.:4780-5905
www.cerrajeriaurgencias.com.ar
Atención las 24 Hs. ¡Precios Bajos! Especialidad en Autos y Casas.

Cerrajeria 4704-0910
www.cerrajeria911.com.ar
U R G E N C I A a domicilio $ 50 en capital y gran buenos aires

DeLeonardis S.R.L
www.deleonardis.com.ar
Cerraduras de Alta Seguridad Lagard conocé nuestra línea de productos

Cerrajeria 24Hs 4784-1416
www.cerrajero24hs.com.ar
Consulte precio con sólo llamarnos Toda la Capital Federal en 20min

Segerlock Group
www.segerlock.com.ar
Fabrica de puertas blindadas Cerraduras Mul-T-lock Tel-4218-4680

Puertas De Seguridad
promo.proseguractiva.com.ar/hogares
Sistema Integral de Seguridad Protección las 24hs, Consúltenos!

Cerrajeria 24HsS.fernando

Figure 2.2.: Google: recommendations for security locks

Different recommender systems are used in other Google's services. For example in Google+ and Android devices, friends and VIPs are recommended by inspection of the most often contacted persons. Google Reader offers a cooperative filtering recommender system to find new interesting blogs, using the subscription list of other users that read the same blogs than us.

2.2. According to the network structure

According to the network structure of the system, we classify current recommender systems in (i) centralized, (ii) simple decentralized and (iii) smart decentralized recommender systems.

2.2.1. Centralized Recommender systems

In a centralized recommender system, users send their profiles, annotations, documents, etc. to a central server that calculates all affinities, possible friends or analyzes the profiles of the documents. Figure 2.3 summarizes the building blocks of a centralized recommender system. Almost all current commercial recommender systems are centralized. This way, the owners of the recommender system can easily monetize the recommendations and charge for them. Amazon, Pandora, FilmAffinity, MovieFinder, Google and most of the recommender systems that were analyzed so far are centralized.

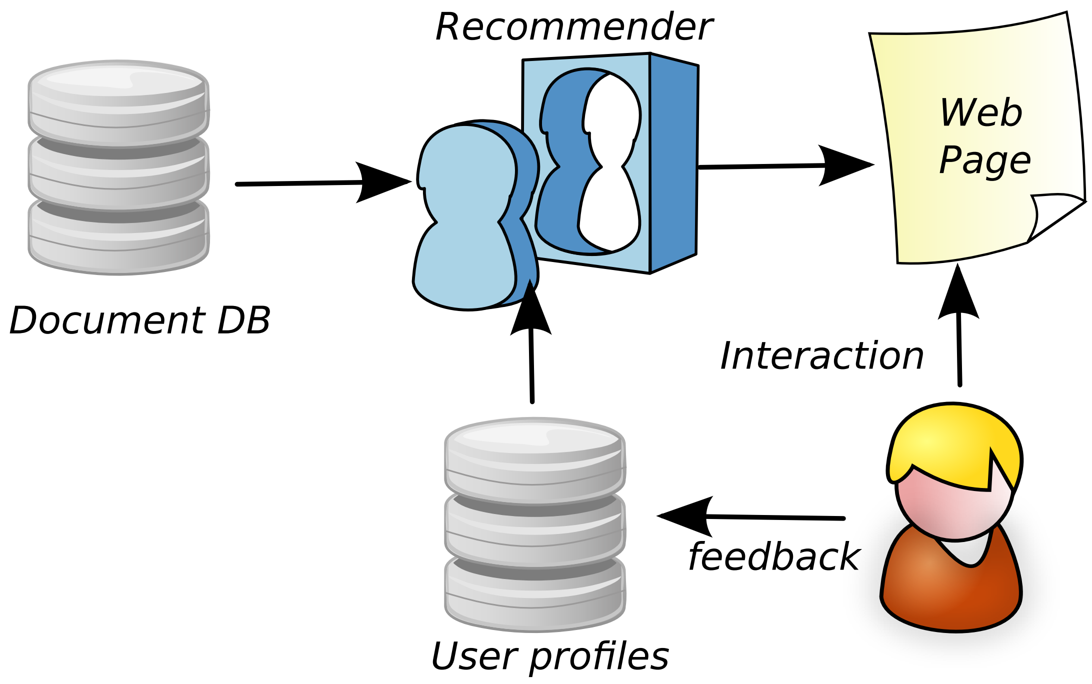


Figure 2.3.: Centralized recommender systems

Centralized recommender systems share these characteristics:

- A central database of customer and/or document profiles exists somewhere in the system. This database controls every piece of data in the system. As a consequence, the quality of the recommendations is limited by to the correctness of the similarity metrics, and not by the network structure.
- Customers address to a well-known, central entity to get recommendations. This is the only entity in the system that issues recommendations. This entity usually plays both the roles of profiler and recommender.
- Centralized recommender systems show many security issues based on the fact that users have to trust on the central node. For instance, customers expect that the central node provides only suitable recommendations, without any malicious bias, and that the central node won't make a misuse of the private information that customers provide to receive a personalized recommendation.

Bawa et al. [19] introduced a system that indexes shared documents inside an intranet. Nodes identify and index their shared documents and send this list to a central repository. Customers searching for a list of keywords contact the central repository and get a list of nodes that store documents that include these keywords. Next, these nodes are contacted to confirm that they store the desired document. Nodes maintain databases of often used keywords and nodes, so they can be accessed without the need of the central repository. Furthermore, there is a recommender system that works in parallel to the searching service, and users can assign a recommended document to some keywords. Bawa et al. proved that such system scales up to one thousand nodes in an intranet. The main problem of this approach is that the central register of the system makes it prone to attacks of denial of service, and provides with little scalability. Finally, authors do not took into account any security issues, since this system is deployed inside the walls of an intranet and they assume that every node can be safely trusted.

SENSE [31] was a recommender system in a decentralized network to distribute documents with a precomputed global database. The existence of this central database is the reason to classify SENSE in this category. SENSE classifies documents in three different groups: *socials* are those inside the cluster of friends explicitly declared by the user; *spirituals* are documents recommended by other users that are alike the searcher; and *global* are documents recommended by every node in the network, regardless of their affinity. Searches and affinity are calculated from the overlap of keywords in the common documents.

2.2.2. Early decentralized networks

In a decentralized or P2P network, all nodes are equal and there is not any service run only by a single participant. This way, documents, profiles, customers and recommenders are organized, managed and indexed in several nodes of the network, maybe replicated databases and structured in a decentralized way. There is not a single point to get a recommendation, and customers may ask to their neighbors for some interesting documents. In early decentralized networks, users are organized and linked together according only to the necessities of the network, in order to maintain the structure and function of the network.

The P2P decentralized structure is suitable for social networks thanks to two characteristics of these networks, (i) the fact that every node is equal to any other node of the network (peer) and (ii) the lack of a traditional server. Those are a very convenient characteristic to cope with the challenge of replication of data and services. In this regard, any node of the network may offer any service or resource. Besides, the dynamism of users in mobile environments is well covered by networks where none has a special significance to maintain the network.

Early decentralized P2P networks such as Gnutella [71] or Napster included a search service that was based on keyword matching. With simple modifications, keyword matching could be transformed into a profile matching problem, and the mechanism to calculate affinities and similarities that were introduced in previous sections still apply to this kind of networks. The main characteristic of the search problem, hence, is the algorithm used to route messages that contain profiles inside the distributed network. For those decentralized networks that offer an intrinsic service searching, the more remarkable types are:

Flooding. When customers look for a document inside P2P network, they send a message to all the nodes that they link, which in turn forward the message to all nodes that they link, and so on. This is for example the case of the original Gnutella [71].

Hierarchy of nodes. Nodes send a list with their documents to especial nodes named ultra-peers. When customers look for a document, they send the query to one or several ultra-peers of the network. This is the case of some enhanced versions of Gnutella [25, 75] and FastTrack [4]. Often, ultra-peers are chosen among normal nodes when they surpass a certain uptime and have high bandwidth. Next, ultra-peers look for resources among them using a different mechanism, such as a flooding limited to other ultra-peers.

Random Walk. This is a modification of the flooding mechanism. Queries are not sent to all nodes in the neighborhood of a customer, but only to a randomly selected subset of nodes. Many studies showed that this type of search mechanism is efficient for highly popular documents [5, 21]. **Epidemic routing** is an especial form of random walks. In epidemics, the selection of the subset of neighbors that receive the query is not only random, but it is usually based on some characteristics of the neighbor [44, 113]. We will epidemic routing in depth during part IV of this thesis document.

Document identification. In this kind of networks, identifiers are assigned to documents and each node in the distributed network is in charge of a subset of document identifiers. When customers look for a document, they ask to the neighbors that most likely manage its identifier. These ones select their own more likely neighbor and forwards the query, until that the document is found. This is the case of Freenet [27] or Distributed Hash Tables. We will study these mechanisms in depth during parts IV and V of this thesis document.

There is not a single mechanism that solves every problem in search discovery. In this regard, some authors have proposed to randomly choose the algorithm to use in each step of the resource discovery [18]. Unfortunately, different P2P networks have completely different structure, so there are few algorithms interchangeable in the way that the authors of this paper proposed. In order to face this problem, we proposed and developed several mechanisms for searching not-semantic documents in many different networks using a common interface. We called this system the Multiprotocol Service Discovery (MSD, [140, 139, 141, 142]).

2.2.3. Smart decentralized Recommender Systems

P2P search mechanisms evolved to make use of the social structure of the network that they create. In the current decentralized networks, users do not create links only to let the P2P network work as expected, but also to improve the search mechanisms. These smart decentralized systems first organize users in neighbors that share interest, in a way that is similar to collaborative filtering model, and then offer a recommender system on these clusters.

One first attempt to offer recommendations on P2P networks is [117]. Sripanidkulchai et al. presented a system to create links between nodes that share interests in network of the Gnutella type. Authors assume that it is very likely that the nodes that stored the interesting documents in the past will also store new interesting documents in the future. In this regard, each node creates a direct link to the nodes that answered to previous searches and maintain a database with the shared documents. The searching mechanism remains unmodified. Hence, new links do not enhance directly the document discovering mechanism of Gnutella, but improve the time needed to get interesting documents. According to their studies, such an algorithm is able to enhance in a 50% the rate of successful searches and drops the average path to documents from 4 to 1.5 network hops. Besides the good results that this algorithm showed, the proposal faces many problems that prevent an actual implementation of the ideas as-is. First, the algorithm can only be deployed over traditional Gnutella, suffering from the same escalation problem that the flooding mechanism of Gnutella shows. Second, the proposal has no mechanisms to find new groups of interest, relaying on previous successful searches and thus it has a very slow initialization phase. Third, since it is deployed over traditional Gnutella using its searching algorithm this mechanism does not support directly recommendations. Finally, authors of this work did not take into account the security and privacy implications of their mechanism.

BuddyCast [100] used a simple algorithm for searches based on random walks through friends and friends-of-friends. Two drawbacks of this algorithm are that (i) it does not take into account the links that friends share with each other, and (ii) the creation of the initial links is completely manual. Even though according to its authors BuddyCast is able to handle hundreds of users at the same time [99]. Anglade et al. [14] evaluated the previous approach and introduced the comparison parameter in equation (2.7), named the overload fraction f_o. This parameter represents the percentage of interesting documents in the network that an algorithm is able to find. We can formally define it in this way. Being Q the set of total queries in the network, $R_{Epidemics}(q)$ is the set of answers to a query $q \in Q$ that a customer obtained using an epidemic algorithm, and $R_{Exact}(q)$ the actual number of documents in the network that match the query.

$$f_o = \frac{\sum_{q \in Q} |R_{Epidemics}(q) \cap R_{Exact}(q)|}{\sum_{q \in Q} |R_{Exact}(q)|} \tag{2.7}$$

In the Information Retrieval field, this parameter is equivalent to the **recall ratio** [17].

Additionally, [14] proposed a methodology to evaluate clusters of users that share interests. These clusters must be defined prior to the creation of the network. Therefore, the number of possible clusters is static, predetermined and limited. Anglade et al. also proposed the metric defined in [106] in order to calculate affinities between documents. In order to calculate the affinity between two different users, they both must have in common a certain number of documents. If the number of documents that both users have in common is small, the calculated affinity may make little sense. If they have not any common document, the affinity cannot be computed. Hence, joining the social network is a painful process where customers must evaluate hundreds of documents and declare their friends before any search takes place. Even in this case, since it is very likely that the user will interact with always the same set of neighbors, the probability to have separated clusters with the same interests that do not see each other are high.

Schifanella et al. [113] used epidemic algorithms to distribute the assessment that users make about some documents of the system, and their system was able to dynamically create a list of similar users. The algorithm was specially developed for ad-hoc networks and only to distribute document assessments, so it presented the same problem exposed above.

Ruffo et al. [110] took advantage of the small world behavior that most social networks present. This system calculated affinity based on the number of common documents that two users own. If two users have not any document in common they cannot calculate the affinity between them.

Many researches prove that the small world phenomena can be used to improve the performance of searches in P2P networks [26] presents one of these systems based on similarity and small worlds. It works for big networks, maintaining a small degree and diameter of the network. Unfortunately, this proposal is just a proof-of-concept that relays on pre-established links between nodes, and it cannot be used in an actual, dynamic network. [80] is very similar to this project, since it proposes a small world structure of Peer-to-Peer networks to perform semantic searches. Unfortunately, users of this proposal cannot keep their interests private.

2.3. According to the security

As stated before, security is one of our major concerns when designing our system. According to [77], when a recommender system is under attack the possible objectives of the attacker are:

Exposure of personal information. The attacker gathers private information about a single user, such as his likes and dislikes. There is a privacy risk in sending our profiles to any profiler and recommender. This risk cannot be completely avoided if the recommender system is a centralized entity that controls every user's and document's profile in the network. Adding noise and distortion to the personal data is a common mechanism to protect user profiles. On the other hand, distributed systems let higher privacy control of the data that a user sends to the network.

Guessing personal information. Analyzing the data that a user sent to the network is not the only way to learn something about him. For example, an attacker may learn something about a user by means of observing his interactions and/or the interactions of his neighbors. If an attacker is able to guess some information about a node of the social network, or impersonate some user with a fake profile, he may be able to learn some information about the social friends of this node.

Introduction of bias or "push attacks", since the objective of the attacker is introduce bias or *push* specific documents to the customer. The attacker will try to modify the output of the recommender, by means of introducing or removing documents from the final list. This kind of attack can be performed from the centralized recommender, or from any user of the system. For example, a malicious user may insert fake annotations, or fake document profiles into the system.

Damage to some documents/owners or "nuke attacks". This kind of attack is targeted against a single document or user. For example, preventing that the document enters any recommendation sets, or preventing recommendations from a certain user of the network.

Sabotage the whole system. The attacker wishes to put the network down, and no one will receive any useful recommendations.

According to the security that recommender systems offer to their users, we organize current recommender systems in these categories:

Unsecured systems. In these systems, the query of the customer is readable by any participant of the system, the source of the recommendations can be traced down and the final recommendation set is known by anybody in the system. Amazon, Google, and nearly all the systems previously analyzed are unsecured systems. The reader must notice that the fact that communications with a central server are secure does not mean that the recommender system is secure. In this case, the central server knows everything about its users and they have to trust that the central server is not going to misuse that information.

Safe users. In these systems, users are protected and they receive private recommendations and secure documents. Recommendations are protected and only some participants of the recommendation are aware of the process. This is not the same case that unsecured recommendations. In a P2P network where recommendations are managed in a decentralized way, customers may take advantage of an anonymous service to contact the recommender. If profiles cannot be identified and assuming that the initial user cannot be traced down, the system protects the privacy of the users even if profiles are sent in clear.

Safe recommendations. In these systems, the recommenders, merchants, profilers and document repository cannot be identified, or they can prove that they were oblivious of the recommendation and/or document that they were providing. This way, they are safe from a new kind of attack that we will analyze in next sections. We will use the term *deniability* to refer to this property.

2.3.1. Privacy Aware Recommender systems

There are many proposals in the literature to create a privacy-aware recommender system. For example, [23] was one of the first proposals to cope with this problem. In there, recommendations are issued from centralized recommenders that make use of the collaborative filtering approach. Before sending the list of annotations to the recommender, customers remove some selected annotations. Finally, the recommender is able to calculate similarities and recommended documents because it is able to predict to some extend the missing annotations.

Canny [23] described a recommender system in which a centralized singular value decomposition model is created combining encrypted ratings vectors from each user. Attackers cannot learn the original ratings vectors from the protected ones, but users can decide if their original ratings are included in the model using zero knowledge protocols. This way, there is no external entity that has access to the private data of a user, in this case, the vector of ratings.

From this early attempt, tens of different proposals appeared in the literature to protect the privacy of the evaluators and readers. Reference [102] is a recent patent on centralized recommender system that performs searches. Privacy is managed by means of noise addition to the user's data with a given permutation probability. This way, the users that insert pieces of data into the indexer are able to control the amount of personal information in the indexer. The objective of the work is preventing the attackers from identifying the users that inserted grades in an indexer, but they do not try to protect indexers against legal attacks. In this regard, the work grants privacy to the users but it is not interested in providing plausible deniability to indexers and intermediate nodes.

In [20], authors study deniability using "lies" within the user's profile. The mechanism to achieve deniability is by means of Bloom's filters. A query is mapped into a Bloom's filter, and some bits of the filter are switched. In this case, it is possible to tweak the length of the filter and the lying probability to provide deniability to the user, since he could have picked up any possible query that matches a specific Bloom's filter. Authors of [20] do not provide any searching service in their network, and users should meet one by one to calculate their affinity. Furthermore, since there is not any indexer in their proposal, if a merchant leaves the network his documents are lost for the community.

Some authors explored the union of different recommender system to provide a joint recommendation which may be more accurate than any of the recommendations provided by the individual systems. These authors acknowledged the privacy problem that systems must solve while joining databases, and explored cryptographic solutions. In [149], Zhan et al. explore the performance of privacy preserving recommender systems that use homomorphic encryption and scalar products. Zhang et al. [150] use a secure multi-party approach to explore the same system.

2.3.2. Legal protection of the participants in the recommendation process

Glorioso et al. [49] analyzes existing legal threats against content providers. After studying some court decisions, they concluded that, according to the US jurisprudence, intermediate nodes of the communication, even at the network level such as ISPs, are threatened with legal attacks. The main argument for prosecutors is that "making available copyrighted document" also includes document indexers.

Hermoni et al. [60] proposed deniability as the security service that protects nodes in a P2P network against censorship. Their definition of deniability is similar to the one that we are going to establish in this thesis, and the goals of the adversary are the same than in their work. The authors described a system that is able to download a document only if it is known in advance by the reader, since the system in [60] does not provide any search or recommendation mechanism to their users. In addition, they include a strong definition of deniability that is no longer suitable in recommendation systems. Machanavajjhala et al. [84] proved that there is a trade-off for utility and privacy in recommendation systems, as usual when dealing with privacy. In this thesis, we will introduce this trade-off as the "plausible deniability" concept.

Some governments demand that service providers give access to any data that they store after a warrant from their law system. This is the case of most European countries, for example. How to prevent data leakage from servers that users do not control anymore and still performing calculation on data is a main subject of research. Hu et al. [65] assert that plausible deniability is the service that prevents cloud filesystem providers to give away the private data of their clients. They describe plausible deniability as a necessary yet probably undeveloped service for most cloud service providers.

As we do, Mortier et al. [91] found a threat for intermediate nodes and propose "dust clouds", highly dynamic virtual machines on a cloud system and where users run routing nodes for an anonymity service whenever they need. Mortimer et al. claim that this system provides deniability. They take advantage of the monitoring services of cloud systems to let the user prove that they are not the source of any dubious message. To cope with this issue, the application model that we will use in this thesis is similar to the privacy manager for cloud systems proposed in [98]. In their work, the authors propose obfuscation of private data using a module that is local to the client, and the cloud server only processes obfuscated data.

Although they are not targeted to recommender systems, plausible deniability for indexers has been studied in the literature. Strategies for plausible deniable search proposed in the literature usually involves dummy traffic, where nodes issue forged queries into the network that are not really of the interest of the user [28, 104]. This strategy may help the source of the message to protect her privacy, but intermediate nodes and indexers that answer the forged queries can still be prosecuted. An attacker can even forge these messages in order to prosecute a targeted indexer. In any case, this kind of system is not incompatible with our proposal. Indeed, a user really interested in preserving her privacy can use these proposals, while intermediate nodes should use ours. [104] proposes a system where users can deny that they performed a query with some terms to the system. Rebollo et al. defined a social model and they provide deniability as a form of k-anonymity: an observer cannot decide whether the user performed a specific query out of k possible queries. Authors of this work focused their efforts on protecting users and not indexers, and their system was not designed with distributed environments in mind.

3. Overview of the system

In section 2.3 we introduced a classification of security attacks to a recommender system [77]. Lam et al. identified violations against **Exposure** of private data, **Bias** of recommendations (both “pushing” information and “nuking” items out) and **Sabotage** of the system. Due to the nature of the current scenario of convergence of recommender systems, P2P, social networks and cloud computing, we will introduce a fourth kind of violation: attacks against **third parties**. These kinds of violation are attacks against roles that are played by the actors that first appeared with recommender systems. These new actors include recommenders, document providers and intermediate nodes that route messages to the other actors of the system.

3.1. Security Analysis

During this thesis, we are interested in studying the Exposure and violations against third parties. As a side effect, the results of our work can be used to improve the defenses against Sabotage of the information. Protection against the Bias of the recommendation was not our primary objective during the development of our work.

3.1.1. Attacks against user’s privacy

A personal recommender requires some amount of data from the customer. Providing more information may improve the accuracy of a recommendation, as well as increase the exposure of the private data of the user. In [77], Lam explores the attacks to the user’s privacy using different points of view. Lam identifies data that is useful for a recommendation, such as user’s interests, and data that is highly private but (possibly) useless for the system, such as her ZIP code. He also proposes the definition of several privacy metrics:

1. The value of the information that the user inserts into the system (from the point of view of accurate recommendations) For example, the knowledge that a user likes a popular movie may not be as meaningful as the fact that she likes an obscure, fan-made tribute movie by the HP Lovecraft Historical Society. In addition, the importance of these data decreases with the volume of data, and a unit of additional information provides a marginal increase of the system knowledge. For example, in a movie rating service, the overall profile of the user is constructed with the first hundred ratings. When the user has rated 1,000 movies, a new rate can hardly modify the user’s profile.

2. A metric of the risk of exposure of the user (from the point of view of data privacy), and likelihood of publishing sensitive data in the system. Lam also discussed about the nature and amount of information that a user needs to insert into the system in order to get a useful recommendation. Finally, Lam foresaw the usual trade-off between both metrics in a real environment. There are some usual techniques that can be applied to enhance the protection of the user's private data and make identification of the user harder. For example, users could add noise to the ratings of their documents [38], insert forged, random queries between legitimate accesses [104] or join a self-organized coalition of different users to present a joint query to the recommender system [39].

We will study a kind of recommender systems where users participate in a social network and links are created between affine users. Thus, nodes in the neighborhood of a user will share with high probability her same likes, dislikes and interests. If an attacker is able to learn somehow the interest of a certain node, or even push a node with a crafted profile into the system, he may be able to make at least some educated guesses about the interests of his neighbors. This kind of attack is referred as the neighborhood attack [151]. Zhou et al were interested in de-anonymizing a social network by means of analyzing their links, but their analysis and ideas where extended in later works. For example, [103] analyzed the number of nodes that an attacker must subvert to get a significant knowledge about the network. They conclude that the attack was feasible even the current social networks, in the size of millions.

Thus, there is a need to set a bound to the personal information that users push into the network. This limit will affect accuracy that the user expects for her recommendations, and any system that is concerned with the privacy of its users will find necessary a privacy and utility metrics, and resolve the trade-off between these two metrics. Several techniques exist to protect the user's privacy in the literature, but there is lack of research on how these techniques affect the recommendation output. Finally, protection of the private information of a single user is not enough if her likes and dislikes can be guessed from her neighbors' likes and dislikes.

We will address these issues during part II of this thesis document.

3.1.2. Attacks against intermediate nodes

Recently, a new kind of attack against intermediate nodes has appeared. The attackers in this case are legions of lawyers and policemen that put content distributors down using copyright infringement laws.

A new international treaty regarding copyright protection called ACTA is being negotiated at the moment of writing this document. After some secrecy, the consolidated text is now public [127]. Article 2.15 copes with liability of legal persons, and states that "the provisions of this section shall apply to **inciting, aiding and abetting** the offenses referred in article 2.14"[1]. The penalties that this article proposes "include imprisonment

[1]In this citation and other that follow, emphasis is added.

as well as monetary fines". Thus, not only downloading or the provision of a protected document is punished under the ACTA, but also the abetting to the downloading. According to a EU Parliament member, "the lack of transparency of the negotiations has made it very difficult for both civil society and the European Parliament to monitor the drafting process". Despite of this, many European states endorsed ACTA on January 26th in Japan [78], but the treaty was not ratified by the European Parliament on July, 2012 [90]. Many political groups, both inside and outside the EU Parliament, have expressed their concerns that the ACTA treaty could enact new barriers for individual rights, even with massive protests as in the case of Poland.

We believe that *recommending* is dangerously close to *incitating*, which is a punished behavior according to ACTA. This situation is even worse for the recommender system if it includes mechanisms to upload and/or download the protected document.

The ACTA treaty was the beginning of a trend in legislators throughout the world to make people that help or even incite to download copyrighted documents liable of copyright infringement. All over the world, bills with a similar nature and spirit to ACTA are passed to the national Parliaments. This is the case, for example, of the Stop Online Piracy Act (SOPA) and the Protect Intellectual Property Act (PIPA) in the USA, or the so called *Sinde-Wert law* in Spain. These bills have raised lively discussions about the balance between the protection of the rights of the copyright holders, and the open character of Internet where most data is exchanged freely. Most of these bills are currently under heavy modifications, but they have something in common with ACTA: they make companies liable of user's actions if the companies do not react to a copyright infringement notification. Under most legislation these notifications were previously issued only by judicial authorities, but supporters of supplementary controls criticize the slow pace of justice courts. In order to match the fast timings of current economy, they propose that administrative authorities or even IP holders could issue copyright infringement notifications. In some cases, system administrators have a short time to react to these administrative notifications, as short as 5 days in the case of SOPA. These laws allegedly aim to "the worst of the worst" of the document providers, but according to some opinions [62], the "broad and vague definitions" that these laws include are dangerous and may be applied on nearly every site. For example, large action sites and huge social networks will find extremely difficult to monitor every transaction and activity of their users. eBay or Etsy, with hundreds of thousands of fast trades between particular users, cannot pro-actively control the copyright status of the items that users exchange.

Are these threats too exaggerated?

The Pirate Bay is a popular web that indexes files in the BitTorrent network. The Pirate Bay does not provide access to the actual document but only lists a set of addresses that allow potential downloaders to locate the document in the BitTorrent network, outside The Pirate Bay's servers. In April, 2009, the administrators of The Pirate Bay were found guilty of complicity to provide unauthorized access to copyrighted content and sentenced to one year of jail and nearly 3 million euros in damages by a Swedish court [118]. Short after The Pirate Bay's trial, Rapidshare, a popular intermediate node to download documents, handed over personal information about content uploaders from Germany to the courts in order to prevent legal action against the company [6].

Recently, the administrators of the direct downloading site MegaUpload have been put under arrest in New Zealand on behalf of the north-American FBI [131]. According to the FBI, this action "directly targets the misuse of a public content storage and distribution site to commit **and facilitate** intellectual property crime". The FBI accuses the managers of MegaUpload of: 1.- massive copyright infringement and money laundering; 2.- "willfully reproduce and distribute many millions of infringing copies of copyrighted works"; 3.- Creation a business model completely centered on **incentive** the copyright infringement, by means of directly paying to the users that upload the most successful files and enforcing rules that prevent the distribution of long term, private data (such as removing data if it is not access after a short time) 4.- Finally, **not removing copyrighted material** even after they are informed of its existence.

In June 2008, Warner Music, Universal Music, Emi, and Sony pressed charges against Pablo Soto, author of several P2P applications. The companies asked for 13 million euros for unfair competition, since the software developed by Mr. Soto could be used to download documents under the copyright of the reporting companies. Mr. Soto did not upload any protected material to the P2P networks, and he pleaded that he was not able to control the activity of the network users. Pablo Soto was acquitted about the charges of copyright violation, but he had to wait for three years for a final ruling [33]. The high expenses of a lawsuit, the criminal charges that the defendants face and the long time to get a ruling persuaded other site administrators to react to these legal actions. For example, only two days after that the FBI took actions against MegaUpload administrators, dozens of similar sites (Filesonic, Fileserve, Uploaded.to, VideoBB, FileJungle, UploadStation, FilePost, UploadBox, x7.to, 4shared, etc.) either changed their policies or announced a voluntary shutdown [42]. Other direct download sites, especially European companies out of USA soil such as Putlocker or NovaMov, took advantage of the new situation [76].

Finally, other actors acknowledge these dangers and are moving the technologies of their services to safer grounds, at least from their point of view. For example, The PirateBay is going to introduce an additional level of indirection by means of moving their service from torrent distribution and tracking to the indexing of magnet URLs. In a few months, The PirateBay will store only URLs that link to a external, uncontrolled and privately run node in a distributed hash table that stores the equivalent to the metadata that torrent files contained in the past [43]. Currently, PirateBay no longer stores links to a P2P network where the desired file can be found, but links to second level links to identify nodes that participate in the desired BitTorrent network. It is not clear if this indirection could prevent legal prosecution against The PirateBay in the future.

We will address these issues during part III of this thesis document.

3.1.3. Additional concerns: recommendations availability

There are some additional concerns in current recommender systems. We collect them under the "recommendation availability" umbrella, slightly abusing the standard security terminology. This collection of availability concerns are related to the ability that the network has to provide accurate recommendations or even no recommendations at all. We will explore them briefly. Some of these open problems were identified in [122].

- From the point of view of the network operation, there is still a need of defining efficient mechanisms to find similar users in dynamic environments, where users join and leave continuously. Defining the exact meaning of "similarity" and "closeness" between users (i.e. collaborative filtering) or user-documents (content-based filtering) is an open issue.
- Service designers still have to find a way to cope with "gray sheep", users that are not consistent in their opinions and therefore create useless profiles. These users may or may not be aware of their behavior.
- The risk of not predictions available. For example, a low number of similar neighbors increases the risk of not getting predictions from the system.
- The cold-start is a well-known drawback of recommender systems that use collaborative filtering. Proposals that study this problem must cope with the joining process of users, how initial user's profiles are created and how users get their first recommendations. Reference [41] defines this problem and explores some solutions.
- Document sparsity is a problem similar to the cold-star problem, from the point of view of documents. Solutions to this problem should face documents that are not rated by a large number of users, and their description cannot be fully trusted. This problem emerges when a new document is pushed into the recommender service, of a rare document is evaluated only by a tiny subset of users.
- Finally, how to cope with synonyms and items that are actually the same document with slight modifications, and the system models like different documents.

We will address some of these issues (cold-start, finding similar users, low number of similar neighbors) during part IV of this thesis.

3.2. System Requirements

According to the taxonomy of recommender systems defined in chapter 2 and the security analysis of section 3.1, during this thesis we are going to define a fast, distributed and secure recommender system. This recommender system:

- Will use a hybrid approach to provide recommendations, using both document profiles that are assigned based on the contents of the file, and collaborative filtering through the creation of communities of similar users.

- Will use a completely decentralized network, without any node with special functions.
- Will provide a complete recommendation experience, from the creation of the network structure to the final download of the desired document.
- Will separate the roles of system operator, merchant, recommender and document provider in completely unrelated actors.
- Will face the security violations of:
  - Exposure, throughout the creation of a the mechanisms to protect the privacy of the users
  - Third parties, through the provision of plausible deniability to all participants in the network

4. Conclusions

According to our goals of “fast, distributed and secure” recommender system, we will work on a social cloud [24] that represents a social network. Inside, we will create some clusters of nodes that gather users that share similar interests and hence are “affine”. When users join the network, they should find their most suitable cluster according to their interests and then link to other users in the same cluster. There are several proposals in the literature to create this kind of social cloud according to the interests of the users in a decentralized fashion [100, 14, 137].

We will create a cloud system that includes all indexers of documents. This cloud is organized in a “dust cloud” in the sense of [91]. That is, each one of the virtual machines is controlled by a user that instantiates it as needed, and dismisses the machine after a while. The cloud provider maintains N different entry points to the cloud system, as many as clusters of the social network. The users instantiate machines that connect to one of this entry points and to other machines that link to the same entry point. Using this process, there are N different clusters of dusts inside the cloud. We will use these machines as indexers of documents of the recommendation system.

Finally, we will create another cloud system to store the real documents. Some examples of these filesystems on clouds are [134, 116]. This distributed filesystem is in charge of store the documents of the merchants and provide them to the costumers. These documents and the nodes of the system that store them must be secured, in the sense that only authorized users must be able to access them, and distributors cannot be charged of copyright infringement for providing them.

An overview of this model is shown in figure 4.1.

Each of these networks faces different problems from the point of view of security. This document is structured in four parts, and each part addresses one of the identified issues.

Protection of the user’s privacy In our recommender system, the output relies on the interests, likes, dislikes and past history of a user. This information is modeled and contained in a user’s profile. Thus, these profiles contain very sensitive information that not only absolutely identifies a person, but models his private thoughts and views. It is not hard to imagine that most users prefer to keep their profiles in confidence. The objective is not to cover illegal activity but protect the privacy of the users.

The mechanisms to protect the profiles of the users and their privacy are described in part II.

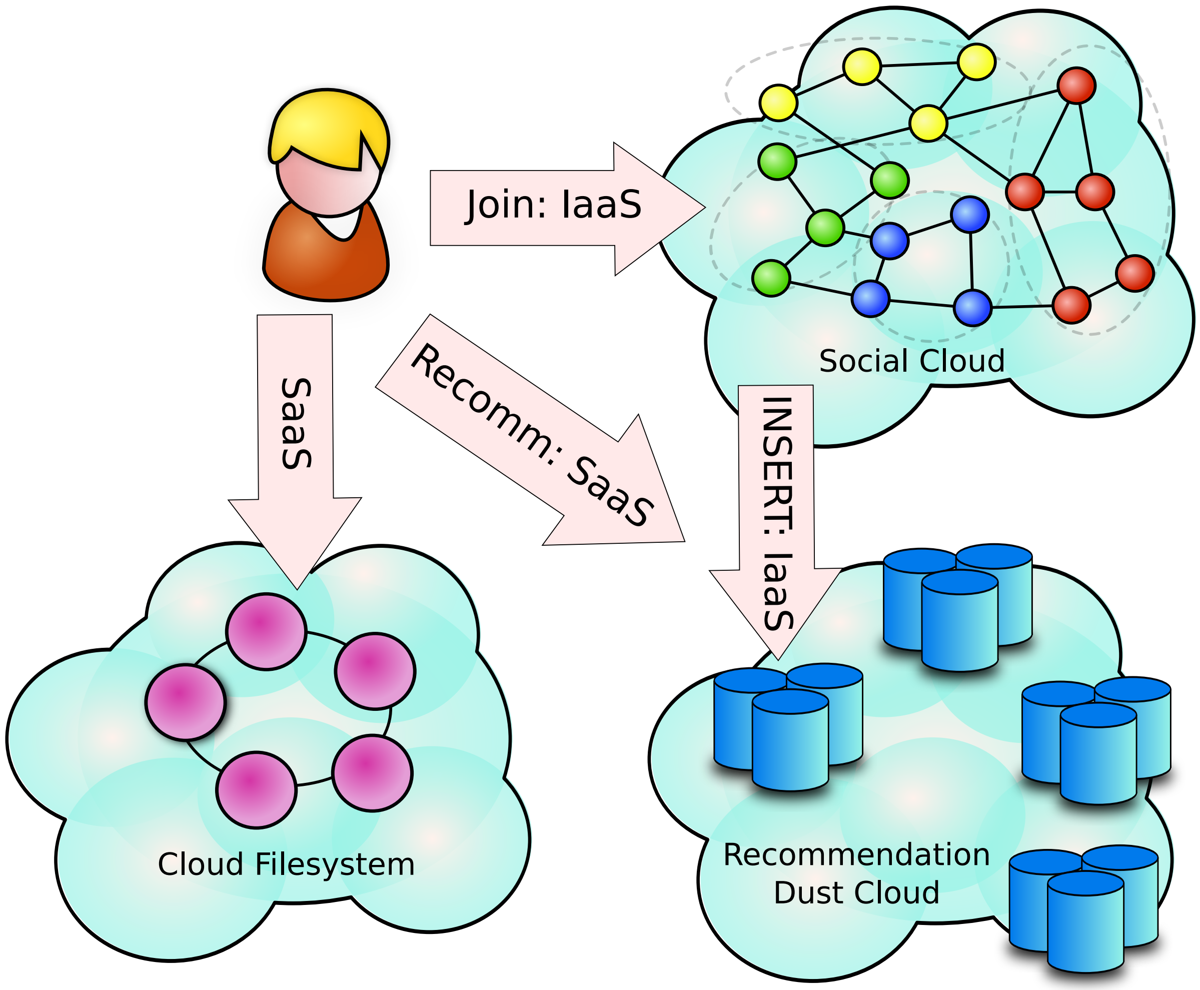


Figure 4.1.: System architecture

Protection of the recommendation source The copyright holders try to stop illegal distribution of copyrighted documents by means of legal attacks. So far, content providers and downloaders were legally reported and prosecuted. These reports have had a variable degree of success, and even couldn't achieve anything in some European countries. As a result, copyright holders shifted their targets. Nowadays, there is a legal threat against people that just aid to identify documents in a P2P network. Event through these people does not store the final document and hence they do not provide any piece of copyrighted material, the indexers of document locations are currently traced down and legally prosecuted.

The protection of the recommendation source is addressed in part III.

Provision of the best recommendation Giving the best recommendation is the main goal of any recommender system. Cold start algorithms, decentralized information and the profile distortion that privacy protection mechanisms introduce are some obstacles in the way to the best recommendation.

In order to protect the source of recommendations, we propose a recommender system that is constructed over a social, not-structured decentralized network where users link each other according to their mutual affinity. Thus, users create clusters of affinity, and they take advantage of their neighbors to find the most suitable document for their interests. The creation of an efficient social network is usually a slow process and very error prune.

This problem is addressed in part IV.

Protection of the content providers The MegaUpload and PirateBay cases of the recent months proved that there is a legal threat against the final providers of protected documents, even if they were not the actors that uploaded the media in the first place. At the time being, legal prosecutors supported their cases on the grounds that content providers have the ability to detect and thus react to the presence of protected media.

In this thesis, we explore a distributed filesystem to spread file pieces and responsibility among many peers. Furthermore, none of these peers must be able to prove that they do not have the ability of identifying copyrighted files and/or file pieces. This system is described in part V.

Part II.

User's privacy protection

Part II: User's privacy protection

In the recommender system that we are going to construct during this thesis, the output of the recommender relies on the interests, likes, dislikes and previous history of the user. Thus, the system must model the user and capture this information in the user's profile. These profiles absolutely identify users and contain very sensitive and private information, since they depend on the users' private thoughts, opinions and views. It is easy to assume that most users will prefer to keep their profiles in confidence.

This part of the thesis models the users' profiles and studies the protection of their privacy. Chapter 5 introduces the social and network models for the creation of these profiles, and defines similarities and affinities between documents and users in an unambiguous way. Next, two different approaches to protect the user's profiles are proposed. The first approach, introduced in chapter 6, distorts the user profile to limit the amount of data that an attacker may learn about users. The main challenge of chapter 6 is the protection of the user's privacy while allowing calculations about their similarity. The second approach, explored in chapter 7, proposes a zero-knowledge protocol to calculate whether or not two users are affine without leaking any information about their profiles.

5. Models and assumptions

According to the definition of a recommender system that was introduced in section 1.1, the main objective of a recommender is to select a subset of all the available documents that are interesting to a specific customer according to some definition of "interestingness". The decision of whether a document is interesting or not depends on the specific customer. For example, a document that is principal for an individual may be dully or even disgusting for another person. As a consequence, the output of the recommender, or the subset of documents that it provides at the end of the process, depends on the specific data that the customer inputs in the system.

In this chapter, we define the model of our recommender system and how recommenders decide that a document is of the interest of a customer. We will model users, documents and queries as vectors that represent their profiles. Next, we define a mathematical function over the vector space to model the proximity or affinity of the different profiles in the system.

5.1. Selecting documents

Different recommender systems have different mechanisms to decide if a document is interesting or not for a certain user. We explored these differences and used them to classify recommender systems in section 2.1.

For example, Amazon is a collaborative filtering recommender that analyzes the past history of the users of the store, identifies histories that share some items and recommends those documents that are in one user's history but not in the other. A pure implementation of this model does not take into account the characteristics of the recommended document, like genre, title or author, and only the presence of the document in another client's history produces a recommendation.

A completely different paradigm is used by the music recommender Pandora. In this system, the profiler analyzes some intrinsic characteristics of the documents, like their musical genre, duration, author, lyrics, instruments, or rhythms; defines some proximity metric using these parameters and recommends new documents that are similar to other documents that the user liked in the past. In this model, the recommender system does not take into account the opinion of other clients and only the objective, internal classification of the documents based on their parameters is used to produce a recommendation.

The problem of creating a social model for recommendations can be described in a formal way [10]. Let U be the set of customers and D the set of all possible documents. Let g be a utility function that captures the guessed rates of documents for each user,

$g : U \times D \to \mathbb{R}$, where $\mathbb{R}$ is a totally ordered set of guessed document ratings. The recommender problem is expressed in this way. For each $u \in U$, choose the document $d_u \in D$ that maximizes the utility for this particular user u. That is:

$$\forall\ u \in U,\ d_u = arg\ max_{d \in D} g(u, d) \tag{5.1}$$

There are two main issues in this formal definition: (i) the selection of a function g such as the guessed ratings are close enough to the ratings that the user would assign to documents, and (ii) the ability of the recommender to maximize this function.

5.2. Social model: user's likes and dislikes

One of the first approaches to create recommendations takes into account the information that the system has about its clients. In these systems, users classify documents in two simple categories, i.e. "I like it" and "I don't like it". This way, the user profile is the specific classification of documents that the user made. Then, user's profiles are sent to the profiler that compare them according to a specific affinity metric and identifies subsets of users that issue consistently similar rates. Finally, a recommendation for a user u is a document that a similar user v rated as "I like it", but u had not rated yet. This simple mechanism is the base of the model that we propose in this chapter.

In a network N there is a finite set of unique documents $D = \{d_1, d_2, \ldots, d_m\}$, referred as documents according to the model defined in section 1.3. These documents may be described by means of metadata, fields or internal inspection in a way that it is possible to define a *bag of words* for each of them, as proposed in [85]. The concept of *bag of words* is a useful model to analyze documents that include natural language. In this model, a document is represented as an unordered set of words and any additional information such as grammar or contextual data is not considered. The bag of words is usually fed into a text classification scheme. An example of this work-flow is a spam identification engine that uses Bayes classifiers under the assumption that the presence of individual words is mutually independent. Next, we will show how a Bayes classifier with word independence uses the "bag of words" model.

We define a system with a set of classes or categories $C = \{c_1, ...c_N\}$, a set of all possible words W, and a set of documents D where each document $d \in D$ is modeled using a bag of words $d = \{w_1, ...w_d | w_i \in W\}$. The system defines a probability function $P(wi|c_i) : W \times C \to [0, 1]$ that captures the frequency that a single word in used in the context of a category. This function is usually created through training document sets. A Bayes classifier is a function $b : D \to C$ that assigns to a document $d \in D$ the most likely category $c \in C$, where probabilities are calculated using the Bayes theorem and the chain rule over the bag of words of the document, as in Equation 5.2,

$$\begin{aligned} P(C|w_1...w_d) &= \frac{P(C)P(w_1...w_d|C)}{P(w_1...w_d)} && (5.2)\\ &= \frac{P(C)P(w_1|C)P(w_2...w_d|C,w_1)}{P(w_1...w_d)} && (5.3)\\ &= \frac{P(C)P(w_1|C)P(w2|C,w_1)P(w3...w_d|C,w_1,w_2)}{P(w_1...w_d)} && (5.4)\\ &= \frac{P(C)P(w_1|C)P(w2|C,w_1)...P(w_d|C,w_1,w_2...w_{d-1})}{P(w_1...w_d)} && (5.5)\end{aligned}$$

Now, real implementations of Bayes classifiers on documents often simplify this calculation assuming word independence. This assumption states that words are conditionally independent and hence pairs of words are not analyzed. In this case, $P(w_i|C,w_j) = P(w_i|C)$ and equation 5.5 is simplified to 5.6,

$$P(C|w_1...w_D) = \frac{P(C)\prod P(w_i|C)}{P(w_1...w_d)} \tag{5.6}$$

Thus, under the assumption of independent words, it is possible to use the bag-of-words concept to classify documents in categories. This concept has been successfully used, for example, in most spam classifying systems or in practically all web-search engines.

The *bag of words* model does not analyzes semantic relationships between words, and the quality of the classification suffers if different words has the same meaning (synonyms) or the same word have different meanings depending on the context (polysemia) A step forward this model is the *bag-of-concepts* model [111]. A *concept* extends the *word* to include not only synonyms, but a semantic analysis of the contents of the document to model its semantics. According to [129], it is possible to define an ontology to describe documents in D. This is to say, it is possible to define n semantic orthogonal (i.e., statistically unrelated) categories to classify documents in D. The classification of the documents using this ontology is the *bag of concepts* model of documents. This way, each document $d_i \in D$ has associated a vector within this ontology $\bar{p}(d_i) = \{c_1, c_2, \ldots, c_n\}$. This vector is the **document profile**. In this work we suppose that each component $c_i \in \bar{p}(d)$ is a real number between 0 and 1 that captures the level of interest of the user in each semantic category. We call **social space** $\mathbb{P} = [0,1]^n$ to the vector space that is defined in this way.

In order to simplify our model, we establish that the social space has a constant number of categories, and hence profiles are vectors of a fixed size. This approach of modeling documents as vectors has additional advantages, since it enables the creation of simple, generic operations on profiles to provide additional services. This way, it is possible to propose mechanisms that protect the privacy of the users' profiles or provide a smart management of documents in repositories. These additional mechanisms, which will be studied in the rest of the thesis, do not depend on the size of the vector but

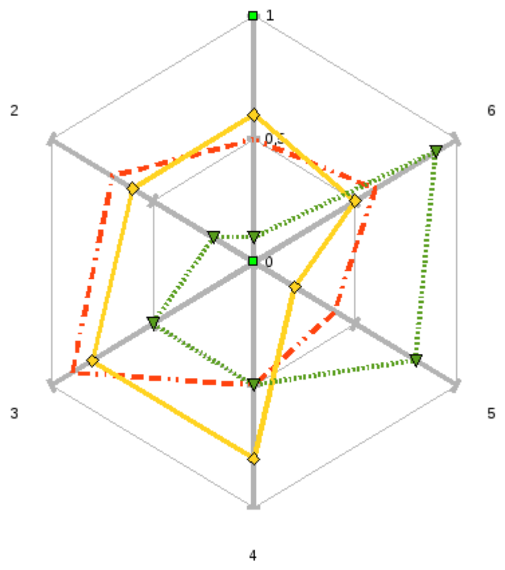


Figure 5.1.: Example of several users' profiles for $n = 6$

rather on the existence of a metric function between profiles. The ideas proposed in this document may be applied to a social model with vectors of a variable size, given that a much more complex metric function is used.

A user u "owns" or "shares" a subset of documents $D_u \subset D$. We suppose that u assigns for each document $d \in D_u$ a pair $(\bar{p}(d), URL(d))$, where $\bar{p}(d)$ is the profile of d and $URL(d)$ a pointer to its location inside an external filesystem that has to be defined. We allow that two different users u and v may have common documents, $D_u \cap D_v \neq \emptyset$. Then, we assign to each user u a profile $\bar{p}(u)$ based on her interests or her documents. In order to keep our model simple, we will assign during this thesis the average of the document profiles that the user owns, as Equation 5.7 shows.

$$\bar{p}(u) = \frac{\sum_{d \in D_u} \bar{p}(d)}{|D_u|} \in \mathbb{P} \tag{5.7}$$

As a consequence, each one of the components of $\bar{p}(u)$ is a real number between 0 and 1 that shows the amount of interest of the user on a determined category of the system. Readers will notice that user profiles defined in this way are also vectors of the social space, $\bar{p}(u) \in \mathbb{P}$.

Figure 5.1 shows an example of three of these user profiles in a social space of 6 different semantic categories using a star representation, as it is usual in the psychology field.

5.2.1. Metrics for profiles

Within this social space, it is possible to define a similarity function: $s : \mathbb{P} \times \mathbb{P} \to \mathbb{R}$. In this document, we will use the cosine distance defined in Equation 5.8 to calculate the similarity between two users. This metric has been successfully used in the past to compare document profiles [106, 85, 79].

$$s(\bar{a}, \bar{b}) = \frac{\bar{a} \cdot \bar{b}}{|\bar{a}||\bar{b}|} = \frac{\sum a_i b_i}{\sqrt{\sum a_i^2 \sum b_i^2}} \tag{5.8}$$

The output of Equation 5.8, $s(\bar{p}_1, \bar{p}_2) = 0$ shows that profiles $\bar{p}_1$ and $\bar{p}_2$ are completely dissimilar, and $s(\bar{p}_1, \bar{p}_2) = 1$ shows that $\bar{p}_1$ and $\bar{p}_2$ are completely similar.

The reader will be aware that the use of similarities in the way of Equation 5.8 provides a search paradigm that is completely different from keyword searches. In a recommender system like the one we are describing, users no longer search for a specific document but rather for some other documents that are similar to the ones that they share. The cosine metric is not the most optimal way of calculating similarity between documents, as shown in the study [87]. However, the cosine metric has an additional advantage that makes it very suitable for our recommender system. Since it is simple enough to compute using only additions and products, it will be possible to use cryptographic mechanisms to provide security and privacy to the users of the recommender system

Given the similarity function in Equation 5.8, and a real number $0 \leq \lambda \leq 1$, we define the affinity function $aff : \mathbb{P} \times \mathbb{P} \times \mathbb{R} \to \{True,\ False\}$ as in Equation 5.9.

$$aff(\bar{p}_1, \bar{p}_2, \lambda) = \begin{cases} True & \text{if } s(\bar{p}_1, \bar{p}_2) \geq \lambda \\ False & \text{if } s(\bar{p}_1, \bar{p}_2) < \lambda \end{cases} \tag{5.9}$$

We say that two profiles $\bar{p}$ and $\bar{q}$ are affine if $aff(\bar{p}, \bar{q}, \lambda) = True$. If $\bar{p}(u)$ is the profile of a user u, $\bar{p}(d)$ is the profile of a document d, and these profiles are affine, we say that d is an **interesting document** for u. We call this metric the **affinity of two profiles**.

The reader will notice that with the previous definition, "interestingness" is unambiguous for a specific metric. Deciding whether or not this metric captures correctly the "interestingness" of a document, or the proper value of parameter λ, falls beyond the scope of this document. Hence, no false positives or negatives are allowed in this model, a recommendation is always correct and, consequently, recall ratio and Equation 2.7 are equivalent. We refer the interested reader to [87].

Searches by users within the network are made through queries $\bar{q}$. Since these queries should be compared with the document and user profiles, they must be in the social space $\bar{q} \in \mathbb{P}$ and should be computed in a similar fashion to document profiles. We assume that a user u will search documents that are related to their profiles $\bar{p}(u)$ in the social space.

Hence, documents, users and queries may be defined as vectors of $\mathbb{P}$. We defined a similarity metric $s : \mathbb{P} \times \mathbb{P} \to \mathbb{R}$ to capture how two different profiles are related, and defined an unambiguous affinity function to check whether or not two users share interests.

5.3. Network model: decentralized system

Social networks are extensively used to share points of view, interests and personal likes and dislikes. If two users share some of their likes, we say that they are affine and they would be interested to link each other in the social network. Indeed, the services that the social networks offer are usually improved if affine users are linked together. When a new user joins a social network, he usually takes advantage of his friends in the real world to create his first links in a manual fashion. It is our vision that the next generation of social networks is about to provide a service to search for other users that are affine but have never met in advance.

We work on a network social network N. Nodes in N are interested in creating a recommender system, as defined in section 1.1. We assume that at any moment, a subset of nodes $A \subset N$ with similar interests is the customer/merchants set of a subset of nodes $B \subset N$ that behaves as the database. Every node in A has links to other nodes in A and at least one link to a node in B. Nodes in B have links to other nodes in B, but not to A. We assume that there is an anonymous routing protocol in the cluster A and an epidemic routing protocol in the cluster B.

We suppose that nodes in B use epidemic routing to distribute and replicate the content that they index. There are many proposals of epidemic protocols for recommender systems [44]. In this kind of algorithms, a query is routed through a well-chosen subset of neighbors that are more likely to give correct answers according to a predefined algorithm. [40] explores the requirements of epidemic protocols: parallel searches, limited connectivity and trust. Some well-known epidemic algorithms that may be used in this scenario are [100, 14, 137]. Indeed, nodes in B save the document description of nodes in A, but they themselves have a profile. In this regard, it is possible to use this profile to create a social network as in [137], but instead of storing and replicating their own documents, they store and replicate the description of the documents of nodes in A. This same epidemic routing mechanism will be used to route the query messages of nodes in A inside the set B.

Additionally, we establish that there is an anonymous routing protocol in the set of customers/merchants A. By "anonymous" we mean that it is not possible for an external observer to decide which one of the nodes in A was the source of a message. One of the schemes that achieve the degree of anonymity that we need is Crowds [105]. In Crowds, given a probability p, a set of nodes A and a destiny $b \notin A$, if a node $a \in A$ wishes to send anonymously a message to b, she sends the message to another node in A, a'. This a' decides with probability p to send the message to the final destiny, or send it to another $a'' \in A$ with probability $1 - p$ and so on.

We define that nodes in B index the data that share the merchants in A. In our proposal, *data* is a pair that describes the documents that nodes in A share, $(\bar{p}(d), URL(d))$, where $\bar{p}(d)$ is the document profile and $URL(d)$ a pointer to the document in an external filesystem. Since nodes in A share interests, nodes in B will index the description of documents that are related to each other.

Nodes in A shouldn't store the profiles of the documents that they share in other nodes of the set A, since all of them share similar characteristics and a malicious node in A may use the information of the shared documents to learn private data of the user profiles [36]. In this regard, all nodes in A agree to store their data in the nodes in B. By means of storing the document descriptions in B, nodes in A may state plausible deniability of the content of queries, as we will see in part III.

Nodes in B are not free contributors. They accept to store data for nodes in A because when they play the role of merchants/customers, they store the description of their own documents in another cluster $C \subset N$. In a real network nodes play many different roles at the same time. For the sake of clarity in the rest of this document, we will model the system such as nodes in A only play the roles of merchants and customers, and nodes in B are recommenders.

5.4. Security model

We consider that any node in the network may be an attacker of the system. That is, the attacker may be the originator of a query, any node in the path between the client and the database or a malicious owner of a database. If the attacker is the database, he is successful if he is able to access the content of a query or the description of the items that he stores. If the attacker is in the middle of the path between a client and a database, he is successful if he is able to identify the source of the query or the database that answered it, or get any information about any of them. If the attacker is the node that sent the query, he is successful if he is able to identify the database that answered the query.

There are at least four ways that an attacker have to gain information about the network: (i) by inspection of the messages that he routes in the network, (ii) his links to other nodes, (iii) confabulation with other users of the network, and (iv) effectively using the services that the network offers. In the case of databases, a fifth source of information is (v) the content that they store.

5.4.1. Plausible deniability

In this work we wish to protect databases owners and intermediate nodes from legal prosecution. From their point of view, *complete deniability* is the ability to deny any knowledge about the content that the database is storing or the node is routing. A trivial solution for this problem is a system where two clients pre-share a key and store an encrypted document in a database. In this case both the database and the intermediate nodes have no means of knowing anything about the contents of the document. However, this approach needs that clients meet and organize in advance, and displaces the encrypted query management from the database to another entity. As a result, we need to define a weaker objective for our work. We say that *plausible deniability* is the ability to deny with high probability the exact content that the database is storing. Even if with this ability the database is able to compute queries and perform searches among its items, then the system works and the database is hardly responsible of the documents that she stores.

We assume that if a process is mathematically *hard*, a node cannot be prosecuted for not doing it. For example, if a database stores data that was modified in some especial way and need to solve a hard function to extract the original pieces of data, we assume that the database owner may claim that the solving process is unfeasible and therefore he is safe against legal prosecution. Examples of *very hard* functions are cryptographic hash functions. A very special class of hard functions is not-injective functions. If two possible pieces of data, one of them arguable and the other completely safe, are transformed to the same item in the database and a random observer cannot decide whether it is one or the other, we assume that this is a hard process.

There is an additional concern in recommender systems. The query that a user sends to the network includes a lot of private information that most people won't likely desire to publish in a database. This personal description may be used by advertising companies to send personalized spam, or to leak the very personal interests of an individual and use this information against her. During this thesis, we will preserve users' privacy by means of limiting the amount of meaningful data that a user needs to send to the network to get a useful recommendation.

It is often argued that we defend the illegal downloading of copyrighted documents. That is, "piracy". Too often, copyright defense is extensively claimed to put down sites that bet for a new business model, or control the freedom of speech of individuals. Most of the complaints against the ACTA treaty go in these lines. The high expenses and long time that a individual user must face in order to defend his case are too overwhelming even if he is not found guilty at the end. In some cases, users are not able to afford them and finally arrive to an agreement with the prosecutor. Furthermore, in a distributed network databases and nodes may help to find or route unlawful and regrettable content just because many of them are run by individuals that has not the ability to control what their machines are doing. We do not support neither piracy nor regrettable content. We support databases that help a P2P network to keep working for a majority of lawful clients, even if that supposes giving service to a few undesirables.

5.5. Outline of the proposed solutions

The amount of interests of each user in the categories of the system is an extremely private information that must be secured. On the other hand, the main mechanisms that the recommender system uses to provide recommendations (i.e., a direct comparison of user's profiles, calculation of the affinity of two users and the creation of clusters of similar users), need some access to the information that the user's profiles capture. This is one more example of the classical trade-off between privacy and utility. In the next chapters, we will explore two different approaches to solve these issues.

- The first approach sets a limit to the information that the user sends to the network, protecting the privacy of a profile by means of a profile distortion that hides the exact amount of interest of users in each of the categories that their profile captures. This distortion takes place using a lineal projection of the original profiles into other social spaces with fewer dimensions. This way, the information about a single category of the original profile is spread between different components of the projected profile. Chapter 6 studies these projections and how they help to improve the user's privacy.

- The second approach protects the privacy of the user by means of preventing any exchange of the user's profile, and two users trying to decide whether or not they are affine only learn the answer to this question. Chapter 7 studies a protocol to calculate user affinities based on zero knowledge systems.

These mechanisms are not exclusive, and the recommender system may use both mechanisms at the same time.

6. Random projections

As described in chapter 5, we model documents, users and queries as vectors of a social space $\mathbb{P}$. These profiles capture the interests of the users and describe them in great detail. As a consequence, profiles include highly sensitive private data that must be protected.

In this chapter, we propose a method to protect the data that a user profiles captures by means of a lineal projection of the vector profiles into a different social space with fewer dimensions. This way, since the projected vectors have fewer components than the original profiles, we have the intuition that they will content less private information about the user, and therefore the exposition of the user to privacy attacks is reduced. Despite this projection, the recommender still has to be able to calculate similarities and affinities with other users and documents, using the metrics that were introduced in chapter 5.

We analyze the proposed method, how useful these distorted vectors are to get recommendations and how much private information the projected vectors store.

6.1. Random projections

To analyze the projection of the user profiles, we will take advantage of two lemmas: the Johnson-Lindestrauss lemma and the undecomposability of random matrices.

Lemma 6.1.1 *(Johnson and Lindestrauss [8]) Given $\epsilon > 0$ and an integer n, let k be a positive integer such as $k \geq k_0 = O(\epsilon^{-2} logn)$. For every set P of points in $\mathbb{R}^d$ exists $f : \mathbb{R}^d \to \mathbb{R}^k$ such that for all $u, v \in P$*

$$(1-\epsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\epsilon)\|u-v\|^2 \tag{6.1}$$

In our work, Lemma 6.1.1 is applied as follows. Given a set of profiles that are modeled as vectors of a social space, it is possible to calculate a projection into a metric space of fewer dimensions that keeps the distances between profiles bounded, and hence their similarities and affinities. In addition, it is possible to configure the error of the final distances to an ϵ suitable for our needs.

The other lemma useful for our developments is Lemma 6.1.2:

Lemma 6.1.2 *(Undecomposability of random matrices [82]) Any $m \times n$ ($n \geq 2m - 1$, $m \geq 2$) random matrix with entries independent and identically chosen from some continuous distribution in the real domain is not two-row decomposable with probability 1.*

Using the first Lemma 6.1.1, we concluded that if we have a social space of n categories, we can define a projection into a space of $m < n$ categories that keeps the distances between profiles bounded. If $m < n$ and the attacker access only to the projected profile and the projection matrix, he cannot reconstruct the original profile since the linear system is undetermined. Lemma 6.1.2 goes a step further. According to this lemma, the malicious user is not only unable to calculate the profile in the original space if $m < n$, but if $n \geq 2m - 1$, he won't be able to calculate any single component of the original profile.

According to these two lemmas, if we use a matrix M to project profile $p \in \mathbb{P}_n$ into a profile $y = Mp^t \in \mathbb{P}_m$ where $n > 2m - 1$, given y, with high probability it is not possible to recover the original components of p and the distances in the projected space are related to the distances in the original space.

In our scenario these projections may be used to preserve privacy of data. The reader will notice that these projections are a kind of data distortion, a technique that has been often used to preserve privacy [38].

6.2. Selection of the projection matrix

Lemma 6.1.1 ensures that there is a projection with these characteristics, but it gives no hint about the actual matrix. In this chapter, we will test three different matrices to create the projections:

- A matrix with random components $m_{ij} \in_R [0, 1]$. We call this matrix M_R
- A matrix with components (Achlioptas [8]):

$$m_{ij} = \begin{cases} +1 & \textit{with probability } 1/6, \\ 0 & \textit{with probability } 2/3, \\ -1 & \textit{with probability } 1/6, \end{cases}$$

  We represent this matrix M_A. This matrix holds proposition 6.1.1, as proved in [8].
- A hybrid matrix $M_H = pM_R + (1 - p)M_A$, where $p \in [0, 1]$

In the simulations that follow, we will use a social space of $n = 200$ (the "200-space"). That is to say, the profile includes the interest of a user in 200 categories. As a first approach, we will use a projection space of $m = 20$ categories (the "20-space"). Given the projected profile and the projection matrix, a malicious user that tries to reconstruct the original profile has to solve a lineal system of 200 variables with 20 equations. There are 180 freedom degrees, and then we can safely establish that the original profile cannot be reconstructed. Under these circumstances, the privacy of the user is preserved, as we will show next.

Since the components of the vector (the interest of a user in a category) are real numbers between 0 and 1, the average profile is $\{0.5, 0.5, ..., 0.5\}$. The maximum distance

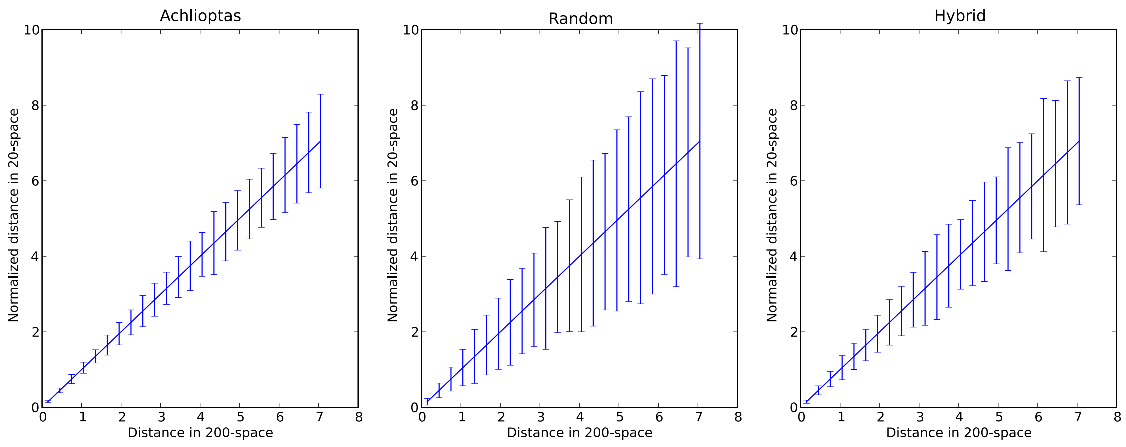


Figure 6.1.: Distances of the projected profiles using M_A, M_R and hybrid matrices

from the average profile to any vector in the 200-space is, using the Euclidean metric, $d_{max} = \sqrt{200}/2 = 7.07$. We will use this maximum value for the distance in our simulations. Since we are interested in how the hybrid matrix behaves, we will use $p = 0.5$. The hybrid matrix approaches the behavior of M_A when $p \to 0$, and the random behavior when $p \to 1$.

Figure 6.1 shows the results of the projection of thousands of vectors using the three types of matrices Random, Achlioptas and Hybrid under study. The horizontal axis shows the distance between two vectors in the 200-space, while the vertical axis shows the average and standard deviation of the final distances between projected vectors into the 20-space. Figure 6.1 shows that there is a linear relation between distances, as Lemma 6.1.1 states. However, the standard deviation of the distance in the projected space increases when the distance in the original space increases. We will study this behavior next.

Actually, there is a scale factor for distances in the m-space. The scale factor is different for each one of the possible matrices, but since there is a lineal dependence between the distance in the n-space and the average distance of the correspondent distances in m-space, this scale factor can be easily computed. In figure 6.1, distances in 20-space are normalized using this scale factor in order that distances in the 200-space and the average distance in 20-space of the correspondent vectors match. The rest of the figures if this document show distances in the m-space normalized in the same way.

Figure 6.1 shows that the standard deviation of the distances in 20-space increases with the distance in 200-space. This is very convenient, since it means that if two vectors are separated a long distance in the 200-space, then the region of possible distances in the 20-space is large. As a consequence, the estimation of the original distance in the 200-space given a distance in the 20-space is probabilistic, and user's privacy is preserved in a certain amount. We can use the standard deviation of the distance in m-space as a measure of the privacy achieved for each one of the matrices. Indeed, the larger this deviation, the larger is the region of distances that a given distance in the 200-space may project. We call this parameter the "uncertainty" of the distance, and it is a measure of the privacy of the proposal.

Proposition 6.2.1 *Given a vector in the n-space $a \in \mathbb{P}^n$, a distance $d_n \in \mathbb{R}$, a projection matrix into a m-space M and a set of vectors $B = \{b \in \mathbb{P}^n | d(a,b) = d_n\}$, the set of normalized distances $D_m = \{d(a \cdot M, b \cdot M)\}$ is a random variable where $E[D_m] = d_n$. We call uncertainty of the distance, $U(n, d_n, m, M)$:*

$$U(n, d_n, m, M) = 2 * \sqrt{E[(D_m - d_n)^2]} \tag{6.2}$$

Figure 6.2 shows the $U(n, d_n, m, M)$ of the different matrices for different values of m and M. The random matrix is nearly independent of the value of m, while the uncertainty of the Achlioptas matrix decreases when m increases. Furthermore, the Achlioptas matrix has much less uncertainty than the random matrix, as expected since it was created with this objective. While a high uncertainty is convenient to preserve privacy, it introduces a higher number of false positives and negatives. The user can control this behavior by means of the parameter p of the hybrid matrix. In every matrix, the uncertainty increases linearly with d_n.

Figure 6.3 shows the uncertainty of the distance $d_n = 1$ when m varies from 10 to 200. The uncertainty is very limited in the Achlioptas matrix, while is large in the random matrix. Furthermore, the uncertainty decreases exponentially when m is increased using the Achlioptas matrix, while the uncertainty is invariant to m in the random matrix. In the proposed procedure of section 6.1, users can set the uncertainty that better matches their interest using hybrid matrices with different p.

Finally, figure 6.4 shows the percentage of false positives and negatives for each of the proposed matrices. For $n = 200$ categories, $d_{max} \approx 7$. As previously discussed, the random matrix has high uncertainty and error. Hence, the percentage of false positives that it shows is high. If the user sets the "affinity" threshold to a distance of $\alpha = 2$, there is a probability of a 25% to find a false positive while using a random matrix. Meanwhile, the errors of the Achlioptas matrix are really low, about 5% for $\alpha = 2$. The limited amount of error is a good reason to use the Achlioptas matrix through the hybrid approach, as figure 6.4 shows.

The fact that this mechanism generates false positives is not a drawback for our recommender system. Even in those networks that make links according to the social distance of the links [80], some amount of randomness should be introduced in order to minimize the network diameter. Additionally, random links do enhance the recall ratio of the random walk searching protocol, as we will show in part IV. As there is only a limited number of false positives in the system, the analysis of this section allows us to determine a minimum dimension for the projected profiles, in order to ensure the privacy of the users while keeping the amount of false positives low.

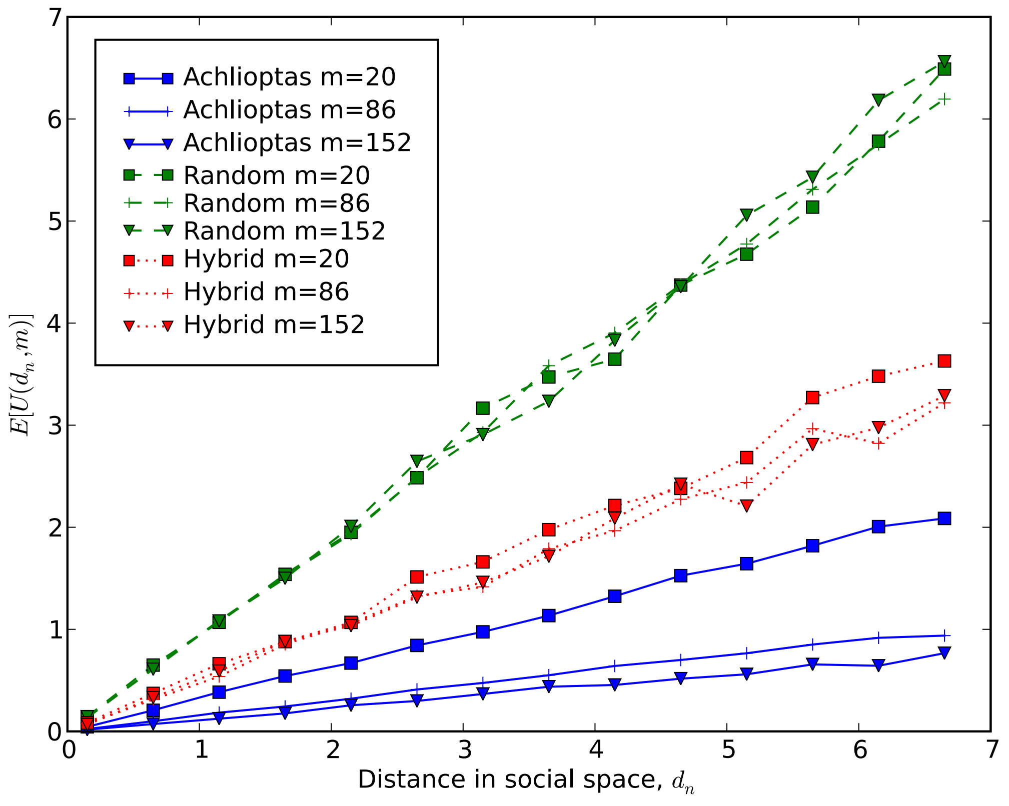


Figure 6.2.: Uncertainty of distances for different matrices

6.3. Triangulation attacks

Random projections show a heuristic behavior. In this section, we will consider the information that an attacker can learn about a profile $\bar{p}$ from the random projection $\bar{p}'$ and the distance $d(\bar{p}', \bar{f}')$ to a profile $\bar{f}$ that the attacker can forge.

One possible attack to the system that was described in this chapter is triangulation attacks. An attacker joins the system and shows especially crafted profiles to localize the real position of a user profile. In our attack model, the attacker can only use the mechanisms that the system provides. Hence, the attacker knows his own profiles, the projected profile of the targeted user and a guess of the real distance between his profile and the user's profile, along with an estimation of the uncertainly as the previous section showed. Since there is not any central authority that authenticates users' profiles, attackers may show any profile that they chose of their convenience. This attack, as described, is a kind of Sybil attack.

In this section, we will show that guessing a user profile from the projected profile and its distance to forged profiles is similar to a random walk in the social space, and guessing the right user's profiles is similar to discovering the profile by chance. In this

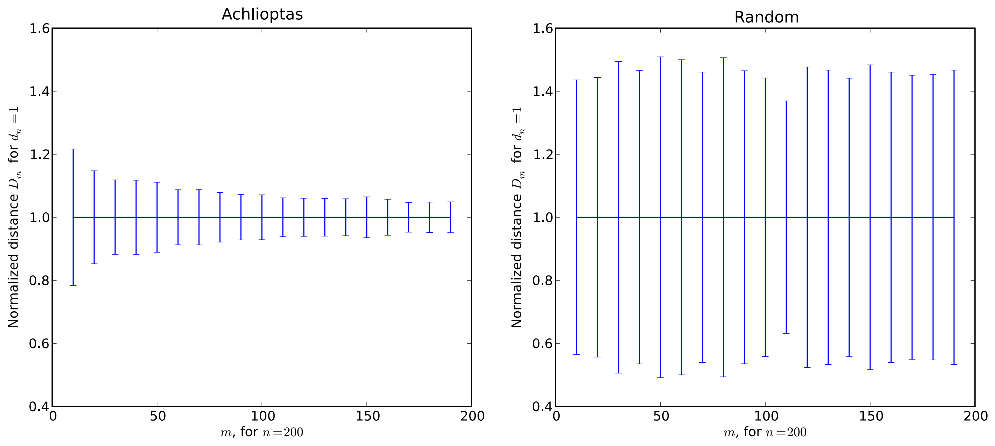


Figure 6.3.: Normalized uncertainty of distances of M_A and M_R

case, this attack is a weak form of a brute force attack.

To model this kind of attack, we define an attacker that makes an additional guess about the target profile. We model this previous knowledge as a sphere in the social space of radius d where the target profile is located. Thus, the attacker forges NA profiles inside this sphere and for each forged profile calculates NT new profiles at the same projected distance than the user's profile. In the figures, we analyze if these calculated profiles are similar to the user profile. For these simulations, $NA = 100$, $NT = 100$ and we will simulate profiles of $n = 9$ different categories projecting to an $m = 3$ space. For the sake of simplicity, only the guess of the first three categories is shown in these figures. If $n = 9$, the average profile similarity using the Euclidean distance in this social space is $d = 1.5$. The profile of the targeted user doesn't change during these simulations. The projection matrix used during these simulations is M_R. This is an implementation of a Manhattan analysis, and figures will include not only the estimation of the target profile but an idea of the uncertainty that the attacker has about the rightness of his guess.

Figure 6.5 shows this attack for an initial distance of $d = 0.01$. The figure shows an estimation of the profile and the uncertainness of the attacker on his guess. As the figure shows, the attacker was not only able to guess correctly all categories of the unprojected, original targeted user's profile, but also he has a low uncertainly about his guess.

When the initial distance increases of attacker and targeted user, the uncertainty of the attacker increases. Figure 6.6 shows the guess of the attacker and his uncertainty when the initial guess of the attacker is $d = 0.1$. Even if the center of the estimation area remains constant, the uncertainty about the real profile has increased significantly.

Figures 6.7 and 6.8 show the guess that the attacker made of the profile when the initial distance is $d = 0.2$ and $d = 0.3$. In the first case, the uncertainly area has grown as much than the attacker only knows roughly a single bit of each category (like/dislike) but not the amount of interest of the user in that category. When $d > 0.3$, uncertainty covers all the social space and the attacker learns nothing about the original profile of the user.

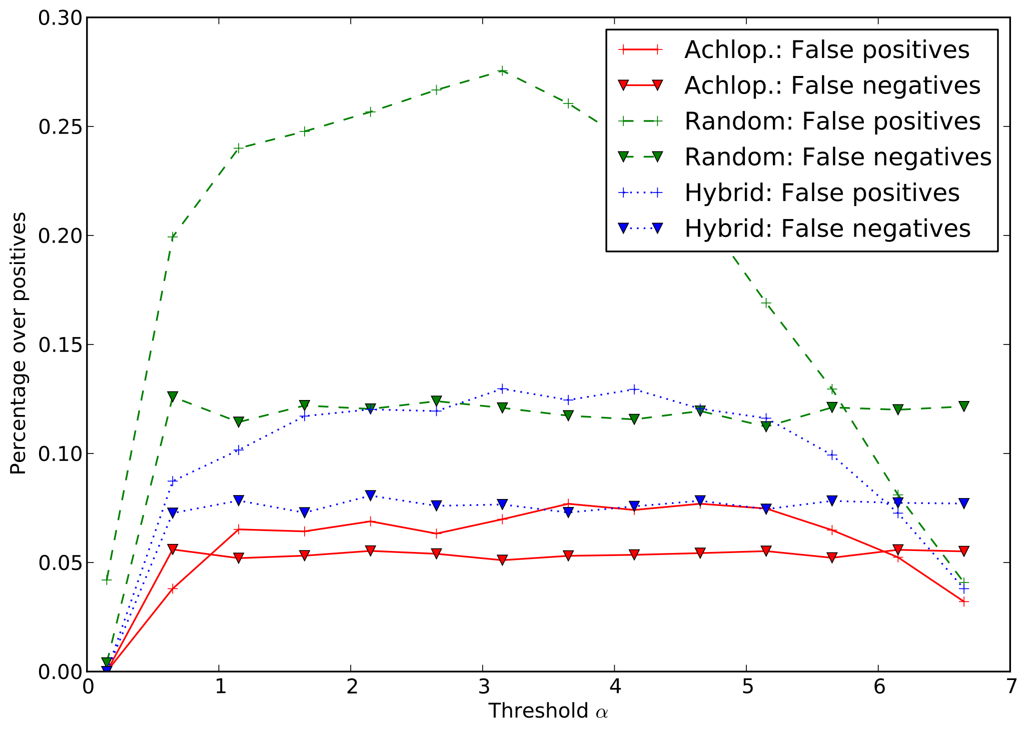


Figure 6.4.: False positives and negatives for several thresholds

Finally, figure 6.9 shows the normalized anonymity lost [35] for the last figures. As discussed, the anonymity lost when the attacker made a good initial guess of the user profile (distance from attacker to user of 0.01) is 0.6. The anonymity lost reduces up to 0.03 when the attacker choses an initial profile at a distance 0.5 units from the targeted user. The reader will remember that in the social space of 9 categories, the average distance of two profiles that are picked at random from the social space is 1.5.

6.4. Discussion

Lineal projections of profiles into spaces of less dimensions let users add a configurable degree of protection to their profiles. Indeed, users can modify the amount of anonymity lost and uncertainty of the projected profile by means of selecting a different projection matrix (for example, modifying parameter p of the hybrid matrix) or increasing the dimension m of the projected space. Especially crafted projection matrices let not only to protect the user's privacy, but still make the calculation of the profiles similarity possible, with a bounded uncertainty.

We explored a kind of attack against this mechanism and concluded that the process of attacking a specific user with forged profiles and triangulation of the original user's

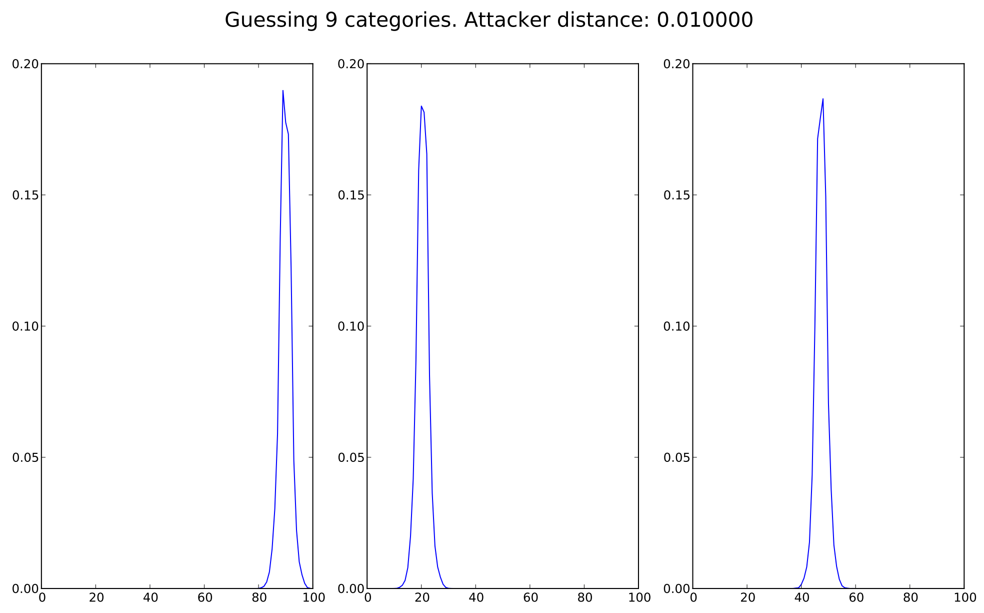


Figure 6.5.: Triangulation attack from $d = 0.01$ (three first categories)

profile need an initial good guess of the user position. If the initial guess of the attacker is further away in the social space, it is increasingly difficult to estimate the position of the original user profile.

There are other attacks to this system that have not been considered. For example, the attacker may trick the targeted user to choose a matrix more convenient for the triangulation attack. Moreover, discovering the profile of a neighbor in a social network where links depend on similarities makes guessing new profiles easier since the attacker knows additional information that can be used to enhance the initial guess profile.

In the next chapter, we will study a different approach that makes triangulation attacks not possible. In the new approach, the output of the process is not an estimation of the real distance, but the answer Yes/No to the question “are two profiles affine?”.

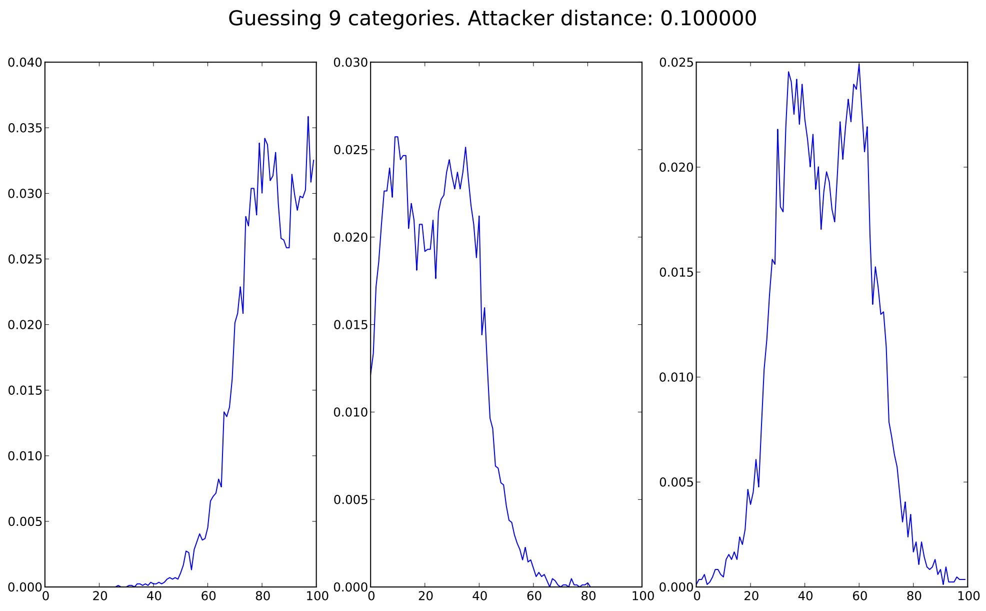


Figure 6.6.: Triangulation attack from $d = 0.1$ (three first categories)

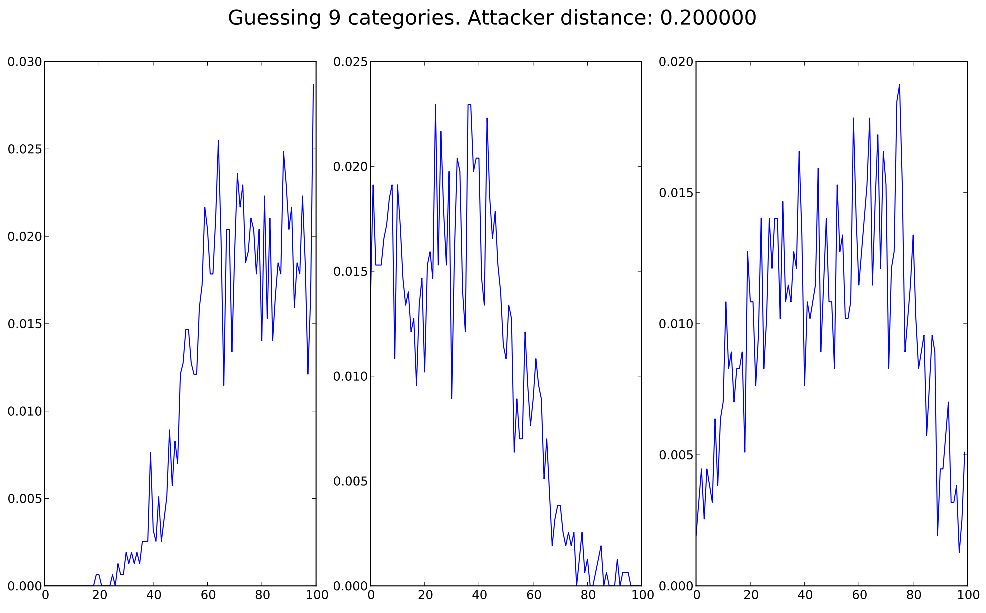


Figure 6.7.: Triangulation attack from $d = 0.2$ (three first categories)

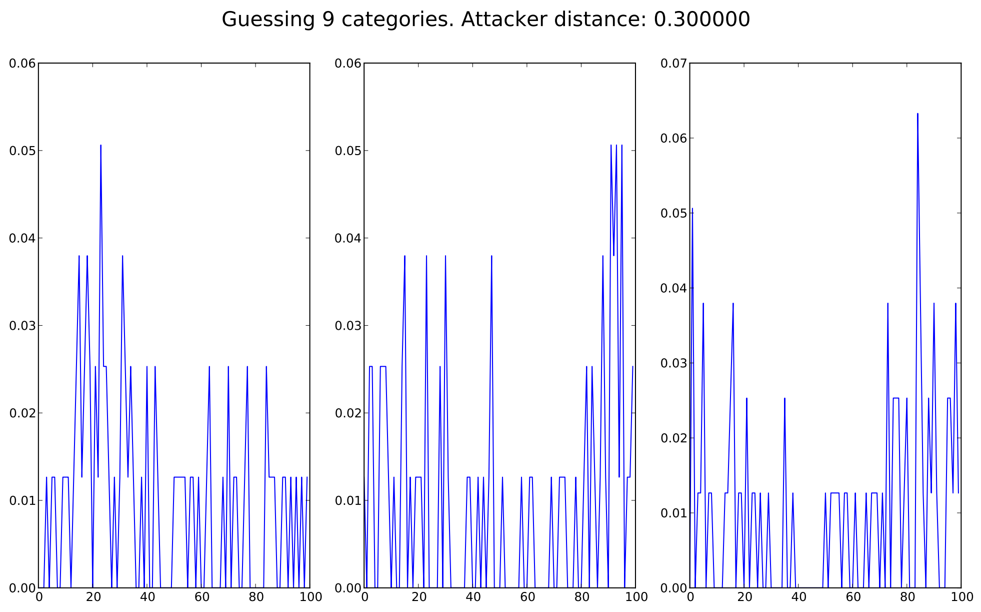


Figure 6.8.: Triangulation attack from $d = 0.3$ (three first categories)

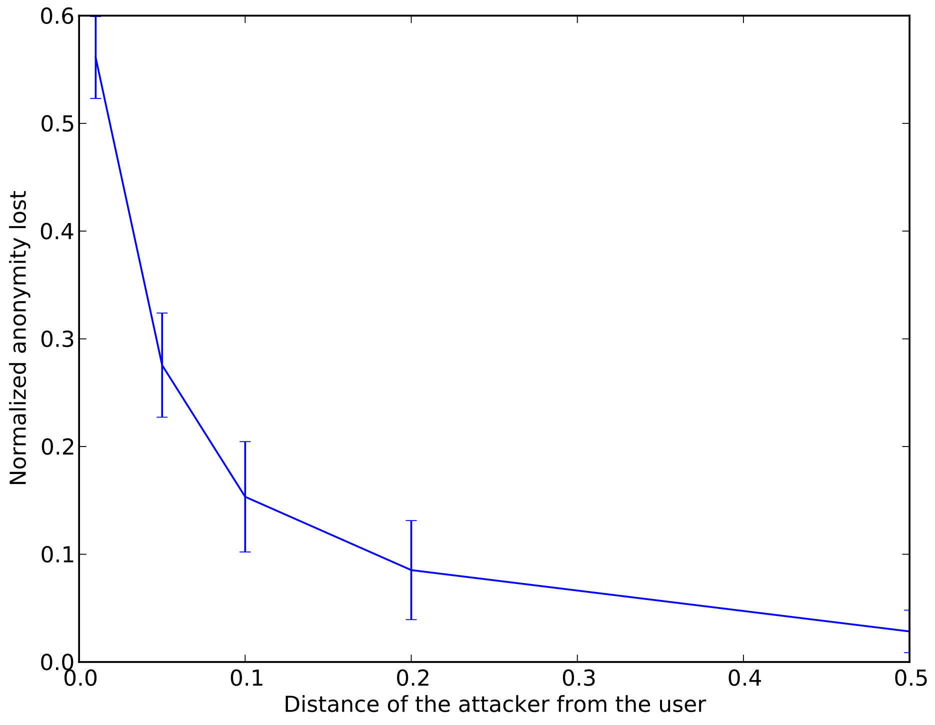


Figure 6.9.: Anonymity lost during the triangulation attack

7. Graph isomorphism

The output of the mechanism that was proposed in the last chapter is an estimation of the original distance between two profiles. As we showed, given the estimation of the original distance, the projected profile and a real projection matrix, the only way for an attacker to learn anything about the original profile was making a good initial guess of the original profile of the targeted user, a problem that is similar to brute force attacks. Unfortunately, the attacker may benefit of a forged projection matrix or especial cases of users' profiles to estimate the real distance from a forged profile and the targeted user.

In this chapter, we will approach the problem of preserving the privacy of the user profiles using a completely different mechanism. We will make use of the graph and subgraph isomorphism problems to develop an alternate mechanism to protect the privacy of the user profiles. In this case, the only output of the protocol is whether or not two profiles are affine, and the protocol does not output any estimation about the distance of profiles in the social space. This way, the privacy of the users' profiles is enhanced and triangulations attacks are not possible. The cost, as we will see, is the need of an increased computation power.

7.1. Models and definitions

7.1.1. Graph theory

A graph is a pair $G = \{V, L\}$ with a set of nodes $V = \{v_1, v_2, ..., v_n\}$ and a set of links $L = \{(v_i, v_j), (v_l, v_k), ...\}$. In this chapter we will only consider symmetric graphs, that is, links in L do not have direction and there are not loops that start and end in the same node. Two graphs G and H are isomorphic if and only if they share the same set of nodes V and there is a permutation $\pi : \{1, 2, ..., n\} \rightarrow \{1, 2, ..., n\}$ such as $\forall\ i, j (v_i, v_j) \in L_G \leftrightarrow (v_{\pi(i)}, v_{\pi(j)}) \in L_H$. A permutation is a relabeling of nodes in V while maintaining the same links. If H and G are isomorphic through permutation π, we write $H = \pi G$.

In complexity theory, it is said that a problem is in NP if there is a non-deterministic Turing machine (NDTM) that is able to solve the problem under polynomial time. A slightly different definition is that NP includes the class of problems that can be proved under polynomial time using a deterministic Turing machine (DTM). An interesting subset of NP is the $NP - Complete$ set. A problem is in $NP - Complete$ if it is NP and any 3 NP problem can be transformed into it under polynomial time. Currently, problems in $NP - Complete$ are considered intractable, since there is not any known algorithm for a DTM that can solve a $NP - Complete$ problem under polynomial time.

Specifically, the problem of checking whether or not a graph G is isomorphic with another graph H is a NP problem, and checking whether or not G is isomorphic with any subgraph of H is $NP - Complete$.

7.1.2. Objective

Two different users, Alice a and Bob b, meet each other in a social network for the first time. They want to know whether or not they are affine. That is to say, given an affinity function d between two profiles of the social space $\bar{p},\ \bar{q} \in \mathbb{P}$, they want to decide whether or not the output of Equation 7.1 is true, for a specific λ that was previously agreed.

$$True : \{d(\bar{p},\ \bar{q}) < \lambda\} \tag{7.1}$$

Alice and Bob do not want the other party to learn any additional data apart from the result of the comparison. This includes not only the profiles $\bar{p}$ or $\bar{q}$, but also any possible estimation of these profiles, their components or the distance $d(\bar{p},\ \bar{q})$ that separates Alice and Bob.

In this work we are going to define a protocol that let both users Alice and Bob check whether or not they are affine for a specific λ, without leaking any additional information apart from the output of Equation 7.1. This protocol is based on the problems of graph and subgraph isomorphism.

7.1.3. Related Work

There are several proposals in the literature that study a scenario similar to the one under consideration in this thesis. Secure multi-party computation using threshold encryption is a common solution. [114] describes a conditional gate that is able to solve complex computations. One of the examples that this work provided as an application of the gate is very similar to Equation 7.1. However, authors limited the definition of interest in the user's profile as binary relations like-dislike, and distance between profiles is limited to the Hamming distance. If we try to apply the conditional gate to a scenario as complex as the one described in chapter 4, the amount of interactions between users to run the protocol with a conditional gate is overwhelming. If we describe user's profiles with hundreds of categories and real numbers, the number of messages that users have to exchange to calculate whether or not they are affine is over several thousands. This huge number of messages seriously limits the application of the proposal to simple profiles and cannot be used in our scenario.

Another approach to private affinity matching is profile distortion. [102] is a recent patent in the field of recommendation systems. The authors described a recommendation system with a centralized entity that performs searches. Privacy is managed by means of noise addition to the user's data with a given permutation probability. This way, the users that insert pieces of data into the indexer are able to control the amount of personal information in the indexer. Even if this is true, we think that private data should not leave the circle of the user. We describe a scenario of a decentralized social network where users do not trust on any servers. The system proposed by [102] cannot be used on a decentralized scenario. [20] also studies a type of profile distortion. The authors considered privacy protection through lying in the profile. The mechanism to achieve privacy is by means of Bloom filters. A query is mapped into a Bloom filter, and some bits of the filter are switched. In this case, it is possible to tweak the length of the filter and the lying probability to provide privacy to the user, since he could have picked up any possible query that matches a specific Bloom filter. This kind of distortion has a serious drawback. An attacker is not able to know for sure whether or not a user is interested in a category since he may be lying, but the attacker can at least make a good estimation. It is our objective not to leak any information of the user's profile, and even estimations will be forbidden.

The graph isomorphism problem was previously used in the security field. [53] describes a generic zero-knowledge protocol that can be used with any *NP-Complete* problem to authenticate a user, and two of the explicitly defined proposals use the graph and subgraph isomorphism problems. In addition to the fact that the proposed protocol cannot be applied to our scenario, [53] does not have into account the approximations to the isomorphism problem that we will analyze in section 7.4. [128] describes a mutual authentication protocol between a client and its access point in a WiFi network that takes advantage of the graph isomorphism problem. Again, the protocol described in this work cannot be applied in our scenario. As far as we know, this is the first time that the graph isomorphism problem is applied for privacy protection.

7.2. Protocol description

In this section we define the protocol that we propose to reach the objectives of section 7.1.2. Given two users that never met in advance, users a and b with profiles A and B, they wish to check if they are affine. Before running the protocol, they agree two real numbers λ and $\gamma(\lambda)$. λ is the value to be used to check Equation 7.1, and $\gamma(\lambda)$ is a configuration parameter that modifies the number of false positives of the protocol. The values of $\gamma(\lambda)$ will be studied in section 7.2.1. In order to simplify the description, parameter $\gamma(\lambda)$ will be written as γ during the rest of the document. Then, both users run the next protocol.

1. Both users a and b agree on a graph $G = (V, L)$, build in such a way that (i) every node $V = \{v_1, v_2, ..., v_g\}$ is a profile in $\mathbb{P}$; and (ii) links in L are random. G is the common input to the protocol. We call g to the number of nodes of the graph. See section 7.4 for additional requisites for G.

2. Users define a clusterization function $c : \mathbb{P} \to V$ of the social space $\mathbb{P}$. This way, for every possible profile $p \in \mathbb{P}$, c assigns the node $v_i \in V$ that minimizes $d(v_i, p)$. The clusters are defined in such a way that the maximum distance between nodes in V that are inside a cluster (the diameter of the cluster) is γ.

3. Secretly, user $x \in \{a, b\}$ calculates the set $HS(X, \lambda) \subset \mathbb{P}$ and $HS(X, 2\lambda) \subset \mathbb{P}$, being $HS(P, r) = \{Z \in \mathbb{P} | d(P, Z) < r\}$. That is to say, the hypersphere $HS(A, \lambda)$ is the subset of profiles in $\mathbb{P}$ under a distance λ from the profile X.

4. Secretly, user x calculates the sets C_x and C_x^2, both subsets of V. Abusing of the notation, we call $c(HS(P, r)) \subset V$ as the subset of nodes in V that are assigned to the hypersphere $HS(P, r)$ through c. Then, users will calculate $C_x = c(HS(X, \lambda))$ and $C_x^2 = c(HS(X, 2\lambda))$.

5. Each user calculates the graphs H_x and H_x^2 as the subgraphs of G that are over subsets of nodes $C_x \subset V$ and $C_x^2 \subset V$.

6. Each user x chooses a random isomorphism $\phi_x : V \to V$, and sends to the other participant $\phi_x(H_x)$ and $\phi_x(H_x^2)$.

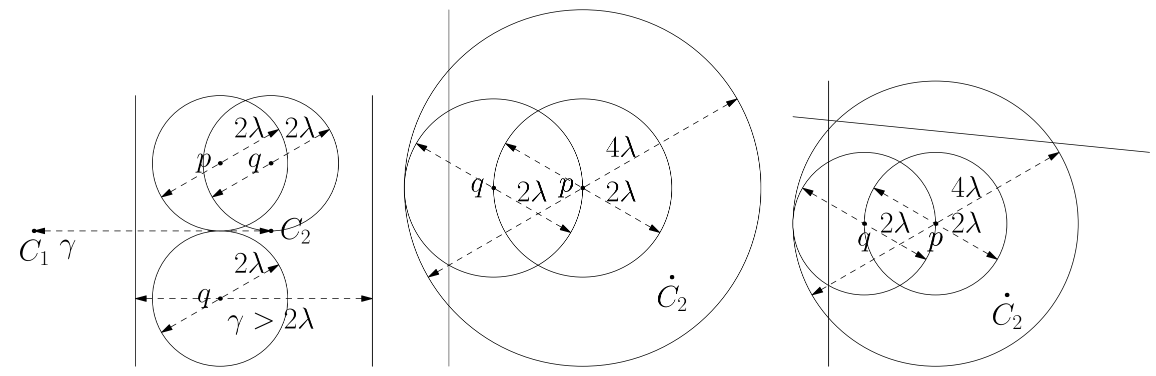
7. If any of the pairs $(\phi_y(H_y), H_x)$, $(\phi_y(H_y^2), H_x)$ or $(\phi_y(H_y), H_x^2)$ is isomorphic, then each user individually accepts $d(A, B) < \lambda$.

Analysis

The main idea of the protocol is that the only data that users exchange are graphs $\phi_x(H_x)$ and $\phi_x(H_x^2)$. These sets are created through an isomorphism of the subgraph of G that is formed with those nodes that are closer to x's profile. If an attacker gets $\phi_x(H_x)$ and he is able to undo the isomorphism, then he would be able to learn the original subgraph and then he can make a good estimation of the user's profile. In this case, the attacker has to solve the problem of subgraph isomorphism which is, as discussed in section 7.1.1, an $NP-Complete$ problem. Even in this case, the legitimate users have to solve the problem of graph isomorphism in the last step of the protocol, which it is a NP problem. In section 7.4 we will discuss the computational viability of this approach. At the time being, the number of nodes in V must be suitable to move the system to an area where the isomorphism problem is solvable, but not the subgraph isomorphism problem.

7.2.1. Bounds to the parameter γ

Next we study the sets C_x and C_x^2. The main idea behind our protocol is that affine profiles that verify Equation 7.1 must outcome the same sets. As we will see, this is not always true in the protocol and a certain amount of errors is introduced. In section 7.2, we defined the parameter γ as the maximum distance between profiles in a cluster of V. This is the diameter of the cluster, and its value should be related to λ. In this section, we will set a bound on the parameters γ and λ to minimize the number of errors. To achieve this objective, we will study some intersection cases between hyperspheres HS_x and clusters of nodes:



(a) Cluster center (b) Cluster border (c) Cluster corners

Figure 7.1.: The three cases for clusters and hyperspheres

Central area of the clusters Figure 7.1(a). If a hypersphere $HS(p, \lambda)$ is greater than a cluster, then any hypersphere will always contain several clusters and the comparison of $C(p, \lambda)$ and $C(q, \lambda)$ does not make sense in the general case. To get rid of this problem, we define that the minimum distance of node in G is $\gamma_{min} = 2\lambda$. In such a way, a hypersphere that is perfectly centered in the cluster does not intersect with any other cluster, and then it is possible to match the sets C_x of the users. In addition, if $\gamma \gg 2\lambda$, then a cluster could house several hyperspheres with a profile even further away than λ, as Figure 7.1(a) shows. In this case the protocol will outcome false positives, since users that are not really affine share the same cluster. This way, it will be necessary to set an upper bound to γ_{max} to limit in the same way the number of false positives.

Borders Figure 7.1(b). It is possible that given two hyperspheres $HS(p, \lambda)$, $HS(q, \lambda)$ and $d(p, q) < \lambda$, $HS(p, \lambda)$ is completely inside a cluster but close to its border and $HS(q, \lambda)$ cuts two or more clusters. In this scenario, a comparison of the two sets of clusters will lead to a false negative. This is the reason why we introduced in the protocol a new set of clusters C_x^2, which are those clusters that intersect a hypersphere centered in p and radius 2λ. Indeed, this hypersphere contains $HS(q, \lambda)$ and then every cluster that is intersected by $HS(q, \lambda)$ is also intersected by $HS(p, 2\lambda)$. Reader will notice that there is no gain in comparing the set of clusters of hyperspheres of radius 2λ, since they will have the same amount of errors that simple hyperspheres. Using this modification, the protocol will give a positive result if any of the pairs of clusters are equal. Like Figure 7.1(b) shows, in order to avoid that two hyperspheres of radius 2λ intersect different clusters, it is necessary that $\gamma > 4\lambda$.

Corners Figure 7.1(c). It is a well-known fact that it is not possible to make a partition of the space using only hyperspheres. This way, the solution that was described in the previous paragraph only works in one direction, and even when clusters that are covered by the hypersphere $HS(q, \lambda)$ are also covered by $HS(p, 2\lambda)$, this one may intersect other new clusters. In this case, the set of clusters C_x and C_x^2 are different, and then the protocol outputs a false negative. We call "corners" to the zones of the cluster where it is not possible to keep the restriction about γ, as Figure 7.1(c) shows. We were not able to find a simple solution to solve this corners, and although it is possible to minimize the effect on the false negatives by means of minimizing the value of γ. In section 7.3, we will study how this effect modifies the protocol outcome.

As a result of these scenarios, we can conclude that we must choose a value for γ larger than four times λ. As larger as we set this value γ, the number of false negatives in corners decreases but the number of false positives in the central area of the cluster increases. Finally, as we will see in section 7.4, there is also a security reason to set γ as low as possible.

7.2.2. Bounds to the value of λ

In order to advance in this section, we need to establish a metric for affinities. We are going to consider the Euclidean distance between user profiles. Even if numeric results do not match using another metric, the qualitative results will.

If we assume that users spread uniformly in the social space, the average profile that we can find is $p = \{\frac{1}{2}, \frac{1}{2}, ..., \frac{1}{2}\}$, and in this case, the maximum average distance to any other profile $\mathbb{P}$ is:

$$\bar{d}_{max} = \sqrt{\sum_n (\frac{1}{2})^2} = \frac{\sqrt{n}}{2} \tag{7.2}$$

According to the bounds that were established in the last section for the elements of G, the maximum distance between two profiles g_i and g_j in G is $\gamma_{max} = 4\lambda$. Hence,

$$4\lambda < d(g_i, g_j) < \bar{d}_{max} = \frac{\sqrt{n}}{2} \tag{7.3}$$

$$\lambda < \frac{\sqrt{n}}{8} \tag{7.4}$$

Equation 7.4 gives a maximum value for the threshold that can be used in dimension n to compare profiles. Values of λ higher than this value do not hold the requisites to run the protocol. For example, in a social network where profiles with $n = 200$ different categories are defined, the highest threshold that users can agree to compare each other is $\lambda = 1,178$.

7.3. Simulation of the protocol

In this section, results of the simulation of the proposed protocol are shown. We define a social space of $n = 20$ categories. We use $\gamma = 0.1$ and define a graph G where sample nodes are spread γ in each direction. To simplify simulations and without loss of generalization, we define clusters as the hypercubes that are centered in each one of

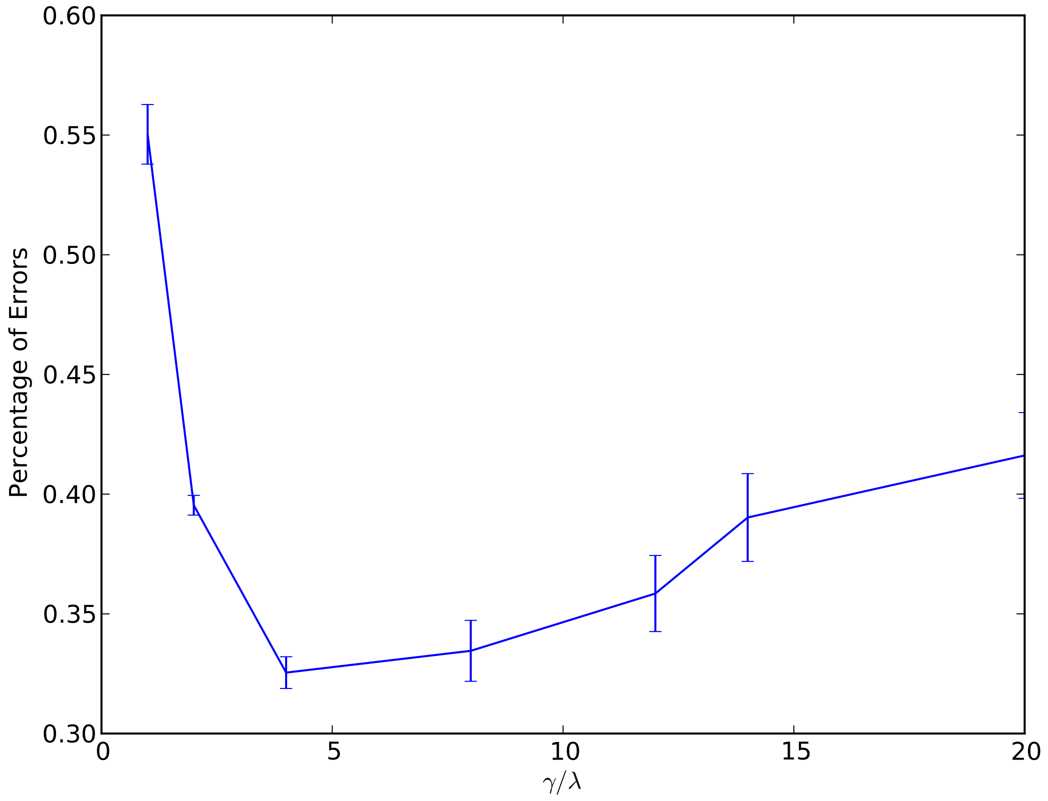


Figure 7.2.: Percentage of errors related for several γ/λ

the nodes of G of side γ. Finally, to check the validity of the protocol it is enough to check whether C_x or C_x^2 of both users match.

Figure 7.2 shows the average and standard deviation of wrong decisions that the protocol takes for several relations γ/λ. The protocol will outcome an error if it estimates that two users are/are not affine but the distance between their profiles is greater/smaller than λ. Figure 7.2 supports the conclusions of section 7.2.1. According to the theoretical analysis, there is a big number of errors if $\gamma \approx \lambda$ due to a large number of false negatives. Simulations show exactly the predicted behavior: if $\gamma = \lambda$, the percentage of errors is over 50%. In addition, we bounded γ to be at least 4λ and foresaw an increasing in the percentage of error for larger γ. Simulations in Figure 7.2 show this behavior.

An interesting issue with Figure 7.2 is that the slope of the percentage of errors is different at both sides of the minimum. We identified two different sources of error in the protocol, false positives and false negatives. It is interesting to put apart these errors, since they have different meaning in the scenario of a social network described in chapter 4. In the real world, it is harder to find affine people than completely strangers: there are lots more people that do not share interests with us than the other way around. In a social network, a lost opportunity to find a real friend is more critical than the

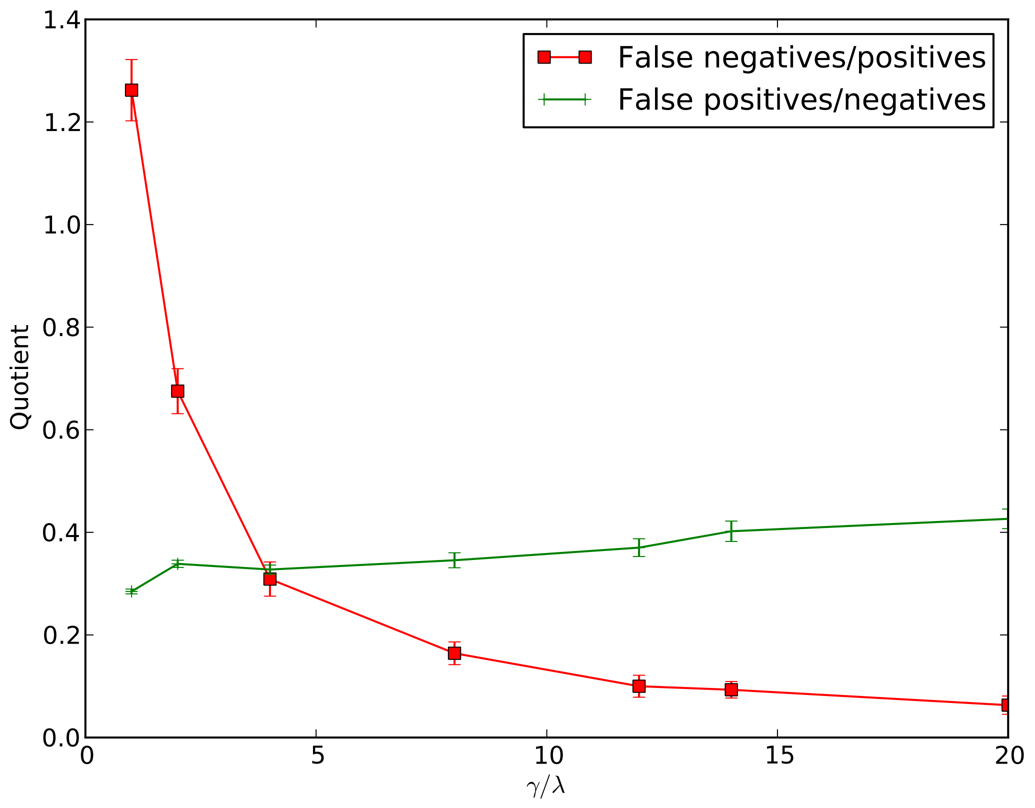


Figure 7.3.: False positives and negatives for several γ/λ

chance that someone is labeled as affine when he is not. Our affine friend-to-be will be lost forever. Therefore, in our scenario, we should be permissive with false positives if this keeps the ratio of false negatives low.

Figure 7.3 shows the relation of false negatives according to the people that the protocol thinks that are affine (positives), and the relation between false positives according to negatives for several values of γ/λ. By inspection of Figure 7.3, a value of $\gamma = 6\lambda$ gets a ratio of false negatives close to 20%, and this ratio decreases exponentially with γ/λ. At the same time, the ratio of false positives against negatives grows with γ/λ, being close to 40% for $\gamma = 6\lambda$. The slow slope of the ratio of false positives is very convenient for our scenario, as discussed before. This shows that we were too conservative when we recommended in Section 7.2.1 a value for γ as close as possible to 4λ. Applications in the real world are likely to show a best behavior with larger γ due to the different significance of false negatives and positives. However, the reader should be aware that, in order to the protocol to be feasible, a small value for γ is recommended. Section 7.4 will discuss this issue.

Even if we are permissive with false positives, they are undoubtedly errors and they will induce a worse behavior of the social network. Figure 7.4 shows some good news

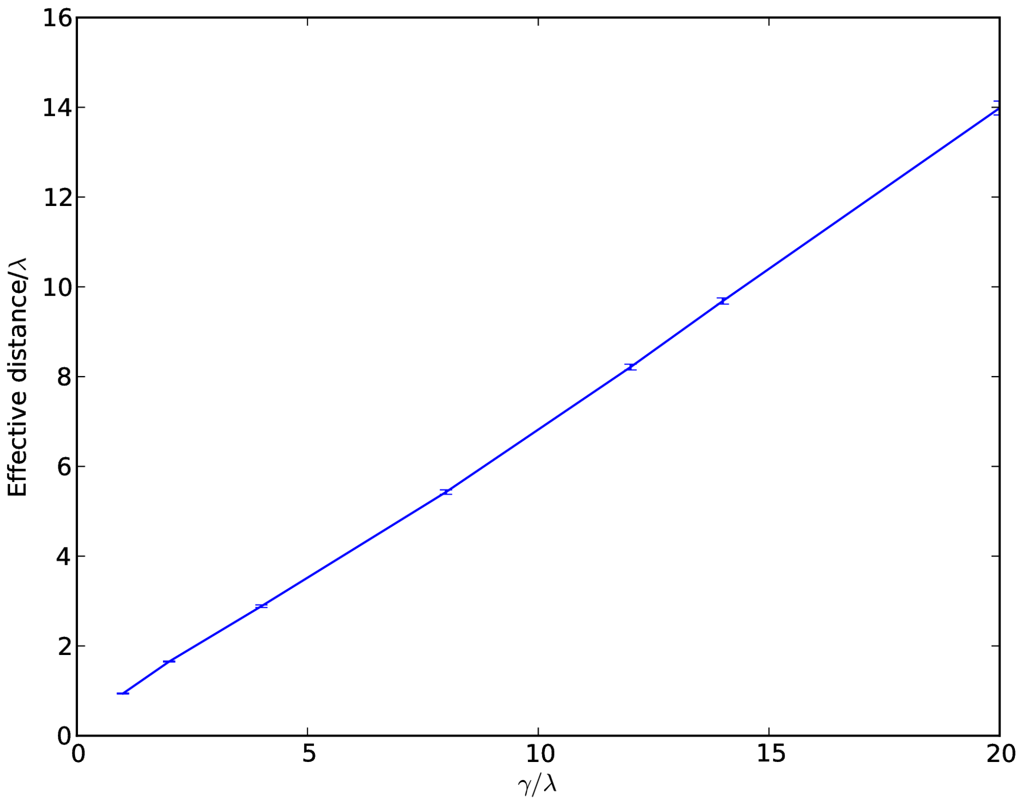


Figure 7.4.: Real distances for several γ/λ

regarding this issue. Readers will remember that a false positive is a profile that shares cluster with other profiles while being separated more than λ. We defined clusters to have a diameter of at most γ. As a consequence, *false* affine people have not really random interests: their profiles are at most γ units from the user. The effect of false positives in the outcome of the protocol is that the affine people that a user is able to find have an effective distance slightly larger than λ, and less than γ. Figure 7.4 shows this effective distance relative to λ for several thresholds λ. In this figure, a value of 2 means that the average distance of affine users according to the protocol is 2λ. As the figure shows, the effective distance grows linearly with ratio γ/λ.

The simulations support the theoretical analysis introduced in section 7.2.1. We recommended a value for γ larger than 4λ, where we find a minimum for the percentage of errors. We can be more permissive with false positives than false negatives, and then values of $\gamma = 6\lambda$ are still safe. Even if these high values exhibit a high ratio of false positives, the effective distance is still bounded to γ. Therefore, the proposed protocol is actually able to compare profiles not under threshold λ, but under a threshold λ' that is between λ and γ. In addition, the use case proposed in chapter 4 has been studied in other works as [137]. In the routing algorithm that is proposed there, false positives help to find a suitable end-point for a recommendation message. Besides, since in our system a false positive has a maximum distance bounded by γ, it is not really a drawback for the system.

7.4. Computation feasibility

Pattern matching inside graphs is a research field more than 30 years old. Despite the fact that the subgraph isomorphism problem is $NP-Complete$ and that there is not a final word about the graph isomorphism problem, both problems have probabilistic solutions in reasonable time. In fact, in order to make our proposal feasible, it is necessary that two conditions are met: (i) both users must quickly decide whether or not the graphs that result from a run of the protocol are isomorphic; and (ii) given one of these graphs, it is not possible to undo the isomorphism and identify the original subgraph of G. The first condition demands that the final graphs are small enough to make the problem of graph isomorphism solvable within a reasonable time, while the second condition ensures that the problem of subgraph isomorphism is not.

[30] proposed an algorithm to solve both problems. Its authors stated that their algorithm has a temporal complexity $O(N^2)$ in the best case, and $O(N!N)$ in the worst, being N the number of nodes in the graph. Furthermore, they show results for the execution of the algorithm to recognize isomorphic graphs, and they can solve the problem for $N = 1000$ nodes in about a second. Besides, in the same work they showed that classic algorithm perform in a similar way. As a consequence, we can conclude that for a number of nodes close to $N = 1000$ the problem of graphic isomorphism is solvable under a feasible time, and then the first requisite of our proposal can be reached if graphs C_x and C_x^2 have a number of nodes of 1000 at most.

We analyze the feasibility of the second requisite next. We assumed that the problem of the subgraph isomorphism is not solvable and then an attacker is not able to identify the subgraph G that is isomorphic to C_a. If he can, then he is able to identify the nodes of G that are affine to a, and then he can make a good guess of a's profile. As described in chapter 4, we wish to apply this protocol to a recommendation system. In order to achieve useful recommendations from the system, we consider that a high number of categories are needed, most likely about several hundreds. We work on a space with $n = 200$ categories. In this case, even if we assume a high value for the minimum distance between profiles in G as $\gamma = 0.2$, the number of possible nodes inside G is huge, $N_{max} = \frac{1}{\gamma}^n = 5^{200}$. A number of nodes as big as this cannot be managed by classic algorithms or by the new proposals as [30], and then we can assume that in our scenario both feasibility requisites are matched.

7.5. Common inputs attack

There is an attack line that was described in current privacy literature that can be applied in this case. If the attacker is able to introduce in G a specially crafted subgraph H

with certain characteristics, the solution of the subgraph isomorphism problem may be simplified. For example, in [16] a method to introduce H into a much bigger graph G is proposed, in such a way that H can be identified even after isomorphisms. If an attacker is able to introduce this kind of subgraphs in G, he can effectively control some areas of the social space. As [16] shows, for this attack to be successful, the graph H needs to have at most $O(\log(N))$ nodes, and each node $O(\log(N))$ neighbors. Authors are able to collect information with as little as 7 nodes pushed in a social network of 4 million nodes. This subgraphs H do not have any special characteristic that let people different from the attacker to identify them, so they can be perfectly hidden inside G.

To solve this problem we propose that the creation of graph G should be controlled by both parties. If G is dynamically created at the moment of the matching, an algorithm to avoid the introduction of arbitrary graphs H should be provided. For example, if both parties agree with a randomized generation algorithm, and a random seed, it is not possible to push a predefined subgraph H into G. Another suitable option is that both parties agree with the creation of graphs G that verified the property of "k-anonymity of graphs". That is to say, given any node and its neighborhood, it is possible to find k nodes in the network that have an isomorphic graph. [151] proposes a graph generation algorithm that holds this property.

8. Conclusions

The recommender system that we are developing within this thesis uses a social network where users create links each other according to the affinity of their profiles. Thus, checking if two profiles are affine or not is a central step of such recommender system.

In order to issue useful recommendations, a user model that captures the user's interests must be defined, along with a suitable metric that compares the similarity between users and documents and decides whether or not a document is interesting for a user. The definition of these user's profiles and an accurate metric are complex and on-going tasks that are being researched transversely in the artificial intelligence and data mining fields.

Chapter 5 introduces a simple model to create profiles of documents by inspection of their contents. From the profiles of the documents that a user owns, we define the user's profile. Finally, queries can be modeled using the same ideas. These profiles are vectors of a **social space**. Each one of the components of a user profile vector captures the interest of the user in one of the semantic categories that the system established. Given this social space, we can define a metric that captures the similarity between the profiles. Finally, given a threshold λ, we defined the **affinity** of two profiles as follows. Two profiles are affine if their similarity is greater than λ.

During chapter 6, the attacker model was an adversary that wants to learn the profiles of the users that participate in the social network. The attacker is successful if he learns some information about profile, that is, if he is able to make a good guess about the interest of the user in some of the categories of the system. Of course, the attack is also successful if he is able to learn the exact value of one single category of the user's profile.

In 6, we explored a distortion mechanism to protect the users' privacy during matching profiles to create communities of affine users. We took advantage of Lemma 6.1.2 to linearly project profiles into spaces of less dimensions. According to this lemma, the attacker is not able to separate the vector elements if the dimension of the projected space is less than half the dimension of the original social space. In addition, we used Lemma 6.1.1 to create a projection matrix that maintains distances of the projected profiles bounded. We explored three of these matrices: Random, Achlioptas and Hybrid. In addition, we explored the amount of private information that the projected profiles store, and analyzed the triangulation attack.

However, users are only safe against passive attackers that follow the rules of the protocol. If the attacker tricks the targeted user to pick a projection matrix that is more convenient for the objective of the attack, or uses the information of the profiles of the neighbors to make good first estimations of the user's profile, triangulation attacks may be possible.

In chapter 7 we developed a protocol to calculate affinity between two users without leaking any information about the users if they are not affine. We defined "affine users" as users with profiles under a distance λ, and we provided mechanisms to tweak the protocol for different values of λ. Throughout simulations we checked that we were partially successful, since the protocol can be configured to bound the number of false negatives as much as necessary, but the overall number or errors increases. Despite this fact, we analyzed that in our scenario a big number of false positives is acceptable if the number of false negatives remains small. Through a theoretical analysis, we proposed suitable values for the configuration parameter γ according to the desired λ. These proposals were supported by simulations.

Part III.

Plausible deniability

Part III: Plausible deniability

In part II, we explored several mechanisms to protect the privacy of a user of the recommendation system by limiting the amount of information that a user sends to the network while allowing the profile matching that recommenders need. Still, there are additional threats for a recommender system. Current legal developments such as ACTA and prosecutions such as the MegaUpload case show that intermediate nodes, indexers and providers can be under attack. We feel that the design of a secure recommender system cannot be complete without considering the legal threats against other actors of the recommender system.

This part includes the proposal of DocCloud, a document recommender system that is deployed on a cloud computing system. This recommender service involves running small virtual machines (PaaS/IaaS) that are controlled by individual users to actually perform recommendations. In this case, we offer legal protection in two different layers: from the SaaS point of view, the system provider cannot be accused of complicity on copyright grounds since he is not aware of the documents that individual virtual machines are recommending; from the PaaS point of view, we protect individual virtual machines and hence their owners by means of hiding the exact machine that answered a recommendation.

Hence, during this work we focus our efforts on protecting all participants of the cloud document recommendation system. Even if the protection of the customers' privacy is not going to be our first priority during this part, some of the decisions take into account the results of other parts of the thesis.

In chapter. 9, we define plausible deniability, explore the models and assumptions of our scenario and present the high level structure of DocCloud. Chapter 10 includes the operation of the recommender system in the cloud and the security analysis of this system. Section 10.3.1 evaluates the contribution of indexers to achieve optimal database anonymity. Finally, section 10.3 analyzes DocCloud according to the requirements described in previous chapters. Finally,

---

9. Protection of intermediate nodes

Moving a recommender system to a cloud structure involves storing document descriptions in the cloud components, sending queries to get recommendations to some entity inside the cloud system and store document and user's profile on the nodes that take part of the cloud. These actions are not only a threat against the users' privacy, but they make it possible for copyright holders to take legal action against the cloud components.

There are two possible approaches to study a document recommendation system on a cloud. If we look at the system as software as a service (SaaS), the cloud provider would be responsible for every recommendation that the system returns. If the cloud manager is only a service provider, in such a way that we are using the platform/infrastructure as a service (PaaS/IaaS) paradigm to separate each one of the cloud components, the individual entities that provide recommendations are responsible for giving access to specific documents. Therefore, from the SaaS or PaaS point of view, the participants of a cloud system can still be accused of distribution of copyrighted documents.

This threat is real, as introduced in section 3.1. Prevention of illegal distribution by means of pure technical solutions has not been effective and copyright holders shifted their targets to the prevention of document distribution throughout legal attacks to document providers and indexers. In April, 2009, the administrators of The Pirate Bay, a popular document indexer, were found guilty of complicity in the provision of unauthorized access to copyrighted content and they were sentenced to one year of jail and nearly 3 million euros in damages [118]. Shortly after The Pirate Bay's trial, Rapidshare, a popular cloud repository to download documents, handed over personal information about content uploaders from Germany to the courts in order to prevent legal actions against the company [6]. Moreover, a new international treaty regarding copyright protection called ACTA is being negotiated at the moment of writing this article. After some secrecy, the consolidated text is now public [127]. Article 2.15 copes with liability of legal persons, and states that "the provisions of this section shall apply to inciting, aiding and abetting the offenses referred in article 2.14". The penalties that this article proposes "include imprisonment as well as monetary fines". ACTA shows a trend in legislators throughout the world to make people that help to download copyrighted documents, and hence cloud providers and virtual server owners, liable of copyright infringement.

9.1. Plausible deniability

The security service that protects indexers and intermediate nodes from legal prosecution is plausible deniability [60]. We introduced this concept in section 5.4.1 of this document. Any indexer or system will achieve **complete deniability** when they are able to deny any knowledge about the content that they are managing. A naive solution for providing this service is using a pre-shared key between data issuer and receiver so that intermediate nodes and indexers cannot access the content. This approach forces issuer and receiver to meet and organize in advance, and displaces the search of documents from the indexer to another entity that shares the key. In order to give a more realistic and practical solution we define **plausible deniability** as the property of the system that protects indexers and intermediate nodes from legal attacks. In the common law context, plausible deniability [132] refers to circumstances where a denial of responsibility or knowledge of wrongdoing cannot be proved as true or untrue due to a lack of evidence proving the allegation.

If indexers and intermediate nodes are granted plausible deniability, they cannot be blamed for the content that they are indexing/routing. Despite the plausible deniability property, indexers still have to be able to answer queries and perform searches to make the recommender system work as a whole.

User's privacy. As described in chapter 5, the query that a user sends to the network includes sensitive data that most people won't desire to publish in a database. In this part of the thesis, we are focus on protecting indexers, but user privacy should be addressed in a complete system. The mechanisms that were explored in the part II must be used in addition to the system described during this part of the thesis. In fact, as we will see, queries and document profiles are going to be protected using the mechanisms of chapter 6.

9.2. System Architecture

We call our recommender system DocCloud. DocCloud involves a social network of similar users, a cloud system of recommenders and a distributed secure filesystem.

Assortative mixing is the property that a network shows when nodes link to other nodes that are similar to them, under some quantitative definition for similarity. Most social networks show an assortative behavior. We work on a social cloud [24] that represents a social network. Inside, there are some clusters of nodes that gather users that share similar interests and hence are *affine*. When users join the network, they identify their most suitable cluster according to their interests. Then, users inside the cluster link each other. We focus our system on N closed clusters, and any user participates only of one cluster. There are several proposals in the literature to create this kind of social cloud according to the interests of the users in a decentralized fashion. Part IV of this thesis will study an epidemic algorithm to create this social network. The interested reader is referred to [100, 14, 138].

In addition, we will define a cloud system that includes all document indexers. This cloud is organized in a *dust cloud* in the sense of [91]. That is, each one of the virtual machines is controlled by a user that instantiates it as needed, and dismisses the machine when the job is done. The cloud provider maintains N different entry points to the

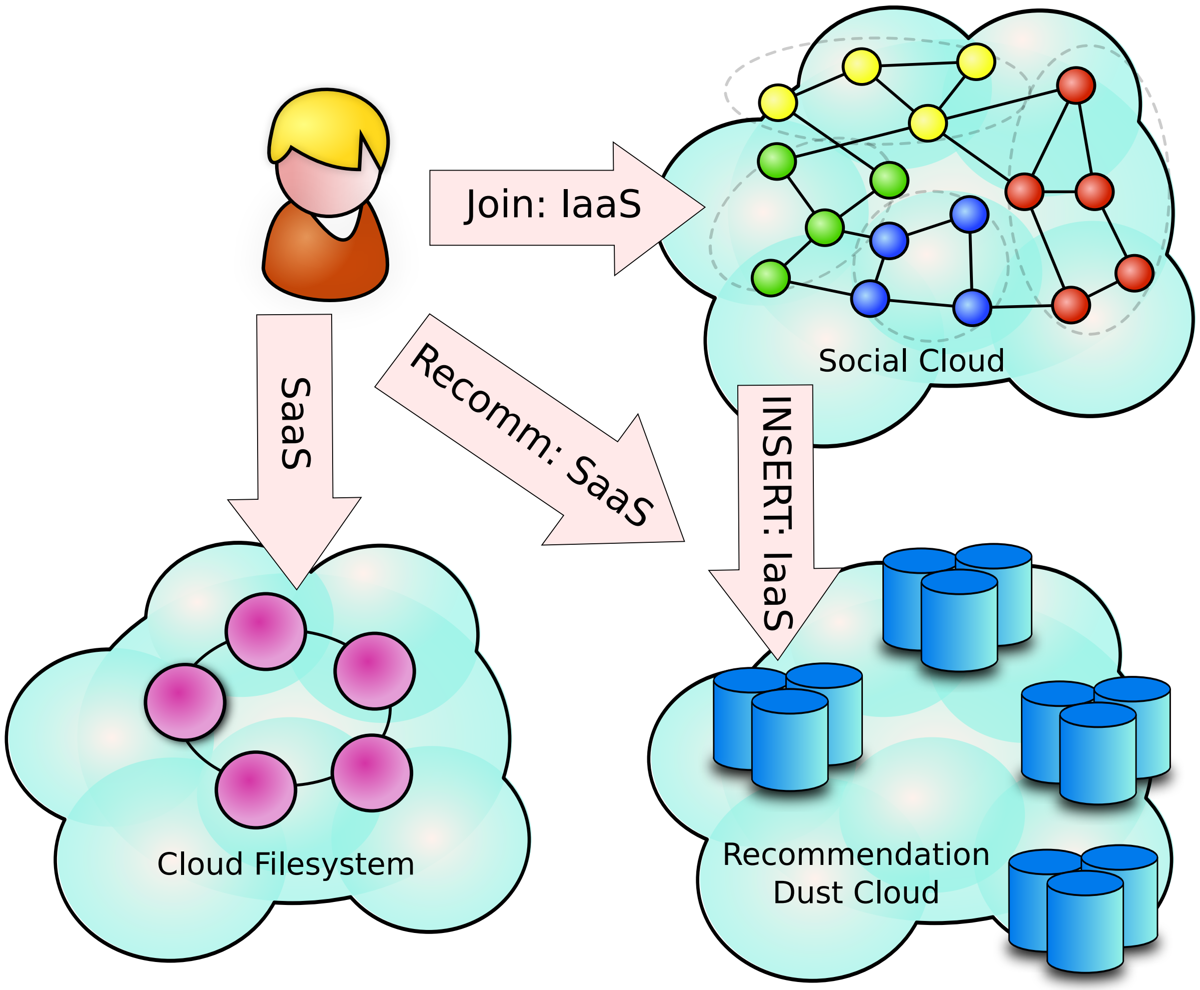


Figure 9.1.: DocCloud: system architecture

cloud system, as many as clusters of the social network. Users instantiate machines that connect to one of these entry points and to other machines that link to the same entry point. Using this process, there are N different clusters of dusts inside the cloud. We will use these virtual machines as indexers of documents of the recommender system.

Finally, our recommender system will use another cloud system to store and distribute the real documents. We will assume that, given an URL, users are able to download a file from this filesystem in a private way. The details about the management of this filesystem will be studied in part V. The interested reader is referred to secure distributed filesystems such as [134, 116].

An overview of DocCloud is shown in figure 9.1.

9.3. Models and assumptions

9.3.1. Document and user's profiles

In order to describe users and documents, during this part we use the same social model that was introduced in chapter 5. Hence, users, documents and queries are modeled as vectors of a social space with N different categories. As explored in chapter II, we can use different mechanisms to distort of even prevent information leaks during the profile matching phase of the recommender process, but these mechanisms are not enough to protect databases against legal prosecution. Indeed, even if databases are not aware of the identity of a user or their profiles, they do know the documents that they are providing.

During this part of the thesis, we will extend the social model described in chapter 5 to include an additional security layer to protect databases. As we will see, profile distortion or zero knowledge protocols don't protect databases against legal attacks and we need to include a cryptographic approach to cope with this new problem. The reader should notice that these mechanisms are complementary to the previously introduced privacy protection technologies, and they can and should be used together.

The cosine metric, as defined in chapter 5, is not the most optimal way of computing similarity between documents, as the recent work [87] shows. However, the cosine metric is simple enough to be computed using only additions and multiplications as we will show, and then it will be possible to use special cryptosystems to protect profiles. We define that a protected profile $[\bar{e}] = \{[e_1], [e_2], \ldots, [e_n]\}$ is a profile where each one of the components was encrypted using an additive-homomorphic cryptosystem such as Paillier [93]. Only the user that knows the private key is able to decrypt protected profiles. We will represent the decryption of this profile as $[\bar{e}]^{-1} = \bar{e}$. Then, given a profile in clear $\bar{a}$ with a known norm $||a||$ and an encrypted profile $[\bar{e}]$, it is possible to make the calculation of a new parameter $e(\bar{a}, [\bar{e}])$ according to equation 9.1.

$$e(\bar{a}, [\bar{e}]) = \sum_{i=0}^{n} a_i [e_i] \tag{9.1}$$

The reader should be aware that operators of equation 9.1 are on the space of the encrypted text, not on the set of integers, and these operations will be calculated by an entity that does not know the private key of the encryption of $[\bar{e}]$. Under these circumstances, given $e(\bar{a}, [\bar{e}])$, only the entity that owns the private key of $[\bar{e}]$ is able to calculate the real similarity as in equation 9.2:

$$sim(\bar{a}, \bar{e}) = \frac{[e(\bar{a}, [\bar{e}])]^{-1}}{||\bar{a}|| \; ||\bar{e}||} \tag{9.2}$$

Next, we make some numbers using the Paillier's cryptosystem. For a social space of $n = 256 = 2^8$ categories and $M = 256 = 2^8$ possible levels of interest in each category, the maximum possible value for $e(\bar{a}, [\bar{e}])$ is $e_{max}(\bar{a}, [\bar{e}]) = [nM] = [2^{16}]$ for a certain profile $\bar{a}$ that holds completely different interests from $\bar{e}$. In a Paillier's cryptosystem, the key bitlength k_p bounds the size of the number that can be encrypted and it is necessary that $e_{max} < g^{k_p}$. In opposition to RSA, there is not any standard for the bitlength of the Paillier's cryptosystem, but we will consider that a bitlength of $k_p = 512$ for $g = 2$ is the minimum acceptable to provide reasonable security. In this case, $e_{max} = [2^{16}]$ is well below the upper limit $2^{k_p} = 2^{512}$. In general, it is necessary that $k_p \leq \log_g(nM)$.

9.3.2. Security model

We will use the security model that was defined in chapter 5. In our security model, we consider that any user of the recommender system or the virtual machines that create the cloud may be attackers of the system. If an attacker is able to learn something about the profiles of the documents or users, or the identity of customers or indexers, we consider that the legal system may assume that these *attacks* are possible. On the other hand, showing that attackers are not successful proves that nodes cannot be prosecuted for not trying to learn information about documents or users.

We consider that users or virtual machines in the cloud may be attackers of the system. The attackers may be (i) the originator of a query, (ii) any node in the path between the customer and the indexer, (iii) a malicious owner of an indexer or (iv) the cloud provider. If the attacker is the source of a query, he is successful if he is able to identify the source of a recommendation. Second, if the attacker is in the middle of the path between a customer and an indexer, he is successful if he is able to identify the customer, the specific indexer that answered the query, or gain any information about their profiles. Third, an attacker acting as an indexer is successful if he is able to access the contents of a query or the exact profiles of the documents that it indexes. Finally, the attacker that acts as the cloud provider is successful if he is able to identify the identity of the indexer that answered a query, the source of the query or the contents of any of the messages that the different participants exchange.

In particular, during this part of the thesis we are going to focus on providing these security services:

- Indexer plausible deniability. Indexers of documents should not be aware of the profiles of the documents that they are serving. Anyway, they still should be able to provide correct recommendations. This way, an indexer cannot be prosecuted for complicity in finding copyrighted documents.

- Cloud provider plausible deniability. Similar to the indexer deniability property, the cloud provider should not be aware of the recommendations that the system is providing.

- Oblivious routing: users that route messages in the network should not be aware of what they are routing. In this regard, nodes that assist in identifying documents by means of routing queries cannot be accused of abetting copyright infringement. Systems that use onion routing are able to provide this service. However, nodes that take part of an onion-routing are oblivious to the content of the message that they are routing. To be effective, nodes that route queries of a recommender system must decide the next hop according to what they believe will provide higher efficiency. In the next chapter, we will study how to tackle with this trade-off between efficiency and security.

- Indexer anonymity. Customers do know the query that they send to the system and the results of this query. If they are able to identify the indexer that answers a query about a sensible document, they may accuse the indexer for abetting the download of the document.

We do not consider a global attacker that is able to observe all transactions in the network. Furthermore, we do not aim to offer protection against denial of service attacks. In any case, since data will be replicated inside the several databases of the network, this kind of attack is much harder to perform. In this part of the thesis, we do not consider the possibility of identifying malicious nodes, but a system that includes identification and marks the malicious users will improve greatly its security.

10. DocCloud

As introduced during the last chapter, we are building a recommender system where nodes are organized in an unstructured way according to their interests. That is to say, nodes in the network with similar interests are linked each other with high probability. This kind of social network is used in the literature to provide recommendations [100, 14, 113]. Therefore, nodes join together and create clusters of similar users according to their interests. Inside these clusters, we will define that nodes agree with a shared key and choose a "foreign cluster" to publish and perform searches. These publications and searches are cryptographically secured in order to avoid that databases to know what they are storing.

Figure 10.1 shows this scenario. In the scenario of the figure, nodes in the cluster A share a key K_A, publish documents and send queries to the nodes in the cluster B. Every communication starts in A and ends in B, and nodes in B do not need to be aware of the interests of users in A. In this regard, nodes in A act as merchants and users of the recommender system, while nodes in B have the role of indexers and recommenders of the system.

The reader should be aware the nodes in B, in turn, may act as merchants and users in front of nodes of another cluster C. During this chapter and in order to simplify the description and analysis, we will consider only a single role for each participant of the system.

Our final goal during this chapter is that (i) nodes in A that route the query of a customer learn nothing about the contents of the query; (ii) indexers in B know nothing about the description of the queries $\bar{p}_r$ that they store; (iii) databases do not know whether or not they are answering a particular query and (iv) customers won't learn the identity of the specific indexer that answered a query. In this case, all participants of the communication may plead plausible deniability in case of a legal attack.

In this chapter, we present (i) the architecture of DocCloud, (ii) the proposed algorithms to get recommendations from the system and (iii) the security mechanisms that the system provides. Specifically, we study the properties of plausible deniability and anonymity of the indexers that issue recommendations. This way, indexers can recommend products to the customers while denying with some probability any knowledge about the product that they are recommending, and even denying their participation in the recommendation process.

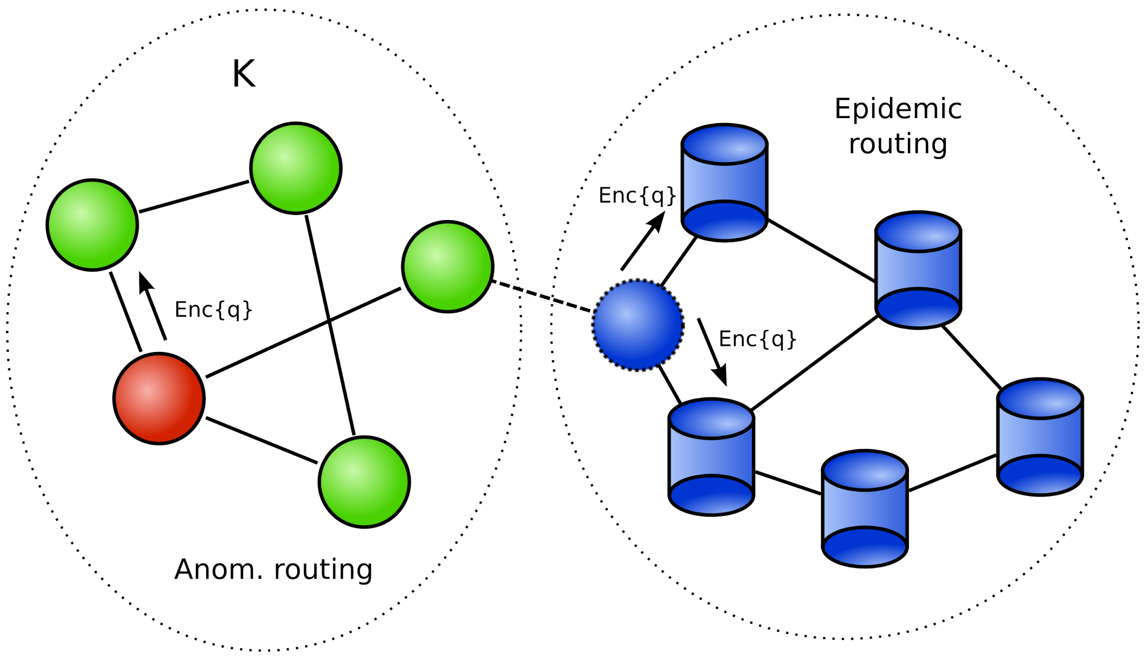


Figure 10.1.: Deployment of the sets of customers and indexers in DocCloud

10.1. Building blocks

In this section we explore the general structure of the system giving its building blocks. In each of the subsections, we explore the involved mathematical tools. In section 10.2, we will join together these blocks to build the complete system.

10.1.1. Key management

Customers will identify and join a cluster of customers with similar profiles. When a customer joins a cluster, a new key must be created and distributed among the members of the group. We propose the key management algorithm of Hernandez-Serrano et al. [61]. This is a Group Key Management (GKM) scheme, which manages the changes of the shared key during the life of a group. The main challenge of a GKM scheme is the secure update and distribution of the shared key among the members of the group.

Hernandez-Serrano et al. aimed to a scalable key management algorithm of large groups of users that minimizes the re-keying cost when the members of the group change. The members of the group are organized in a tree-shape structure with sub-keys in each of the branches. When a user joins or leaves the group, only the correspondent sub-keys are changed. This way, the number of messages to change a key is minimized. The key management scheme of [61] does not need any central node and it is unattended. These characteristics make this scheme suitable for our needs.

The output of the key management scheme produces a shared key that all members of the group know. This key will be updated when a new member joins the group, or an old member leaves. We assume that entities that are not members of the group won't seek actively the group key. That is to say, we assume that attackers may have access to the group key, but legitimate nodes that are not members of the group won't have interest on owning this key.

10.1.2. Random projections

Nodes in cluster A push the profiles of the documents that they share into indexers in cluster B. These profiles are not inserted into the the document indexers databases *as-is*, but they are projected into a social space with less dimensions, as was explored in chapter 6 of this thesis. Therefore, databases index projected profiles and they are oblivious of the real profile that they are indexing. The reader should notice that this is true even if they have access to the projection matrix, and this matrix is something that indexers are not expected to seek actively.

10.1.3. Homomorphic encryptions

Giving a $x \in \mathbb{Z}_q$, we call $[x]_k$ the encryption of x under the key k. We will consider a crypto-system that holds these two homomorphic properties:

- Given $[x]_k$ and $[y]_k$, it is possible to calculate $[x+y]_k$ without knowing k.
- Given $[x]_k$ and y in clear, it is possible to calculate $[xy]_k$ without knowing k.

One of the homomorphic crypto-systems that is proposed in the literature showing these properties is Paillier's system [93].

Hence, if a profile $\bar{p} = \{p_1, ..., p_n\}$ is normalized and encrypted as $[\bar{p}]_k = \{[p_1]_k, ..., [p_m]_k\}$, it is possible to calculate the encrypted distance to another normalize profile $\bar{q} = \{p_1, ..., p_m\}$ using the cosine metric as:

$$[d(\bar{p}, \bar{q})]_k = \star_{i=0}^{n} [p_i]_k^{q_i} = [\sum_{i=0}^{m} p_i q_i] \tag{10.1}$$

This formula is a modified and encrypted version of the cosine metric that was explored in chapter 5. The cosine metric needs only additions and multiplications to be computed, and hence it is possible to devise an homomorphic crypto-system that computes this metric. Despite its simplicity, we judge that equation 10.1 is very suitable for the objectives of this system.

We use this block as follows. Databases index profiles in the form $\bar{p} = \{p_1, ..., p_m\}$. A customer privately creates a secret key k. Then, the customer sends to the database queries $[\bar{q}]$ that are encrypted under k. Using an homomorphic crypto-system, the database is able to calculate the value of the parameter $e(\bar{p}, [\bar{q}])$ using equation 9.1, even if it knows nothing about k and $\bar{q}$. Finally, this parameter is sent back to the user, which calculates the similarity using equation 9.2.

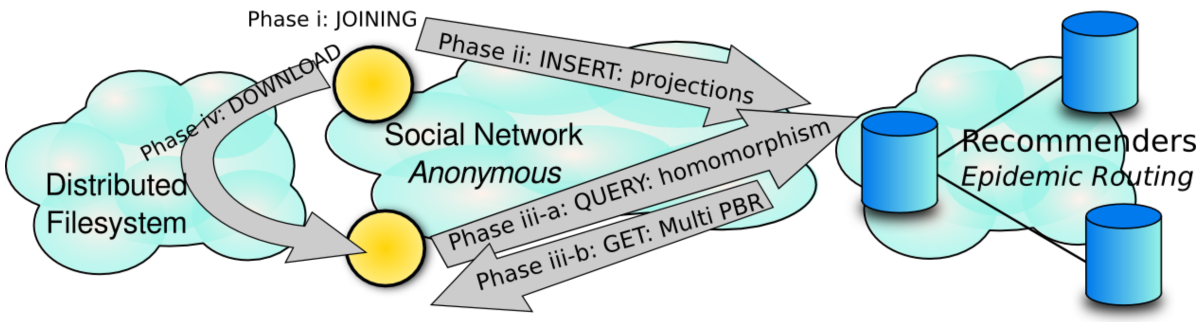


Figure 10.2.: An overview of the mechanisms

10.1.4. Private Block Retrieval

We assume the existence of a Private Block Retrieval (PBR) scheme that works on single databases. In a Private Block Retrieval system, a client is able to get privately j bits from a database, starting from a position i. This way, indexers are able to claim that they do not know which item a specific customer is downloading.

One of these schemes that is suitable for our work is presented in [47]. The complexity of this algorithm is $O(k + j)$, being $k > \log(n)$ a security parameter. The authors of this scheme state that it has the lowest asymptotic communication complexity of the current proposals, and their statement seems not to be challenged in recent papers.

Additionally, the database owner is often interested in adding an oblivious transfer mechanism to their queries. In this regard, a client of the database cannot be able to download a significant piece by means of some forged queries. If clients can only access to a single piece of data with a query, $URL(d)$, then the rest of the contents of the database are safe. The mechanism that is proposed in [47] describes the necessary steps to achieve 1-k oblivious transfers.

In this work we assume that an efficient mechanism that provides both private block retrieval and oblivious transfer exists, We refer to the interested reader to the proposal [47]. For the sake of completeness, we will propose a simplified yet working version of a PBR scheme in section 10.2.3.

10.2. Recommendation System Operation

In DocCloud, we identify four different phases to get a document recommendation: (i) the creation of a social cloud, (ii) the insertion of document profiles into the indexers, (iii) the search of recommendations by the customers and (iv) downloading of the recommended document. Users organize in a social network according to their interests, documents are distributed from a secured cloud computing system that is shown on the left of figure 10.2 and are recommended by the cloud system that is depicted on the right side of figure 10.2.

Figure 10.2 shows the phases of the system, and outlines the mechanisms that we will use in this section during the description of the system.

10.2.1. JOINING: Creation of the social cloud

Users of DocCloud are organized in clusters according to their interests. How new users discover the most suitable cluster of interest and the management of these clusters will be studied in part IV of this thesis. At the time being, we will assume the existence of an efficient algorithm that creates the clustered social network.

After joining to the most suitable cluster, nodes in a cluster A will agree a secret key that includes two items, $K_A = (B, M_A)$, using the block was was introduced in section 10.1.1 The first part of the key is a reference to another cluster of nodes B that will be used to store the profiles of the documents shared by nodes in A. The reader should be aware that nodes in B won't need to be aware of the identify of the cluster A. The second part of K_A is a random matrix M_A that we will use to protect profiles in A, as introduced in section 10.1.2.

An important point to notice is that since users are not authenticated when they join group A, we cannot assume that attackers are not able to get K_A. We do assume that lawful indexers in B are not going to actively look for K_A, and we will see soon that even if they are handed over K_A, they are not going to be able to calculate the original document profiles.

Nodes in B are not free contributors. They accept to index documents for nodes in A because they insert the profiles of their own documents in a different cluster $C \subset N$. That is to say, in a real network nodes are both users and indexers, but for the sake of clarity during the rest of this article we will consider only the merchant/customer roles of nodes in A, and the indexer role of nodes in B.

10.2.2. INSERT: Inserting document profiles into indexers

After users join a specific cluster of the social cloud, they will share some documents with the community. During this phase, user $a \in A$ plays the role of a *merchant*. The first step for a is assigning profiles to her documents, using the mechanisms discussed in section 9.3. For a document d_i, a assigns a profile $\bar{p}(d)$ and publishes the document d under $URL(d)$ in the filesystem in the cloud, as described in section 9.2. Finally, a inserts the pair $(\bar{p}(d), URL(d))$ into a random indexer $b \in B$.

The reader will notice that if indexers and intermediate nodes are able to access these profiles in clear, they will perfectly know the kind of documents that they are providing access to. In addition, they are even able to estimate the user profile just by means of collecting enough document profiles from the same source. Indexers and intermediate nodes need to use these profiles to route and answer queries according to the affinity of the user to the document descriptions that they index. In order to provide deniability for DocCloud, we need to devise a mechanism that hides some of the information of the descriptions but is still useful to calculate affinities between elements in $\mathbb{P}$.

The chapter 6 of this thesis introduced projections into social spaces of fewer dimensions as the mechanisms that provides this service. Two different problems arise in this scenario: (i) whether comparison of profiles makes sense in the projected space and (ii) the amount of information of the original description that is preserved after the projection. Analysis in chapter 6 shows that these properties hold for some projection matrices and $n > 2m + 1$. The reader is referred to chapter 6 for additional details.

In DocCloud, when a user $a \in A$ inserts the description of a document d into an indexer $b \in B$, he sends the pair $(M_A \bar{p}_n(d)^t, URL(d))$, where the first component is the projection of the document profile and the second component is the URL of the document.

The identity of the owner of the document reveals information about the contents of a document, since we assumed that users' profiles are related to the documents that they own, as discussed in section 9.3. In this regard, nodes in A should not publish the pair on their own in node b, but they must hide inside an anonymous cloud. We do not enforce any particular anonymous network, but for the rest of this document we assume the existence of a distributed anonymous network such as Crowds [105] or Dust Cloud [91].

Next, an epidemic routing protocol occurs to distribute information inside the set of indexers from B. The objective of this epidemic protocol is spreading the information of the documents in many different indexers. This way, (i) the availability of the document profiles increases, (ii) nodes in A may contact a random node $b \in B$ to perform queries without compromising the efficiency of the results; and (iii) the possible liability of providing access to a document is shared among different nodes. There are many proposals of epidemic protocols for recommendation system [44, 100, 14, 137, 138]. Part IV will study a specific epidemic protocol for our scenario. Indeed, nodes in B save the document description of nodes in A, but since they have a user profile as well, it is possible to use this profile to organize nodes in B according to their interest, as in [138]. Thus, nodes in B can take advantage of the epidemic algorithms proposed in the literature, but they store and replicate the description of documents owned by nodes in A instead of the profiles of their own documents. This same epidemic routing mechanisms will be used to route query messages inside B.

10.2.3. QUERY & GET: Recommending documents

Our security goal during this phase of the recommender system is to provide indexer anonymity, that is to say, to make it impossible for an attacker to distinguish which one of the indexers is the one that stored a particular answer to a query.

During this phase, the participants view the system as figure 10.3 shows. Even if it is not stated during this section, the reader must take into account that all communications between customers and indexers use the anonymous routing that was introduced in the previous phase. Hence, indexers cannot identify the customer that issued the query that they are currently managing.

We describe a process in four steps. First, customers run algorithm 1. This algorithm issues queries to indexers that run 2, and gets a vector E of encrypted distances. Then, customers decrypt distances and choose the indexes of the documents that are more similar to the queries that they issued. Then, customers download the URLs of the selected documents using algorithm 3. Finally, customers will use these URLs to download the recommended document from an external, distributed filesystem. During this section

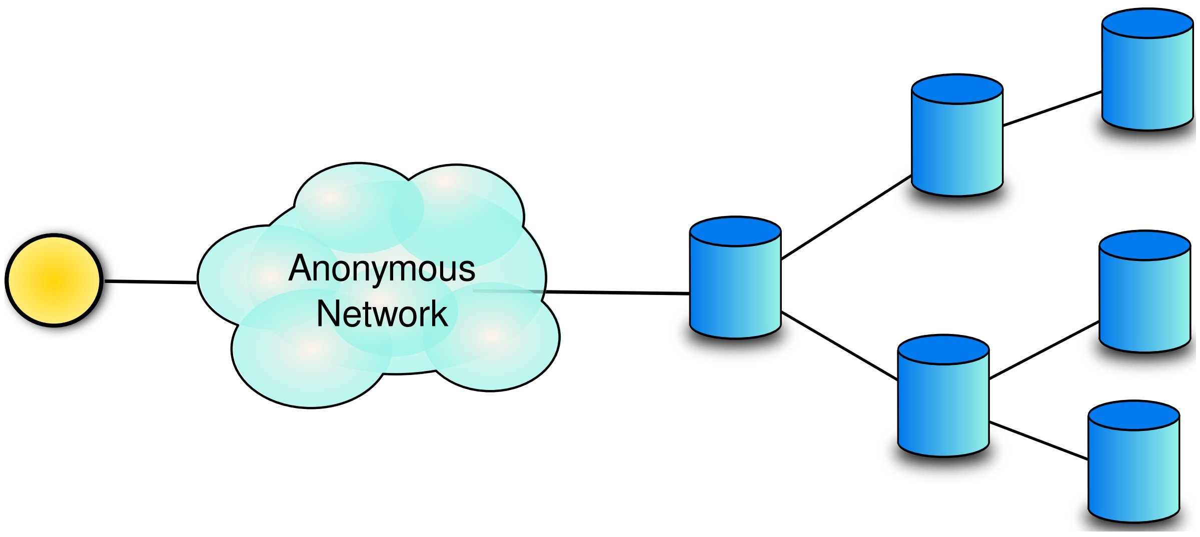


Figure 10.3.: Multiple indexer scenario

Algorithm 1: customer_search_query($\bar{q}_n$, M, K)

```
$\bar{q}_m \leftarrow M\bar{q}_n$
$[[\bar{q}_m]] \leftarrow Enc(\bar{q}_m, K)$
$D \leftarrow DB\_search\_query(b \in_A B, [[\bar{q}_m]])$
**foreach** $[[d_i]] \in D$ **do**
    **if** $Dec([[d_i]], K_{pk}) \leq \lambda$ **then**
        $customer\_download\_pbr(i)$
```

we will look at these algorithms in detail. Even if it is not stated during this section, the reader must take into account that all communications between customers and indexers use the anonymous routing that was introduced in the previous phase. Hence, indexers cannot identify the customer that issued the query that they currently manage.

QUERY document profiles

A user $a \in A$ that searches for a document in an indexer $b \in B$ builds a query $\bar{q}_n$ and chooses a private key K_a. This key is only known by a, which projects and encrypts the query as stated in section 10.1.2 to create $[\bar{q}]$. Next, a sends anonymously $[\bar{q}]$ to a random indexer $b \in B$, which calculates the parameter $e_j(\bar{p_j}, [\bar{q}])$ of every document profile that indexes using equation 9.1, and then creates a vector E_b that contains these parameters e_j. The query is sent to other indexers of the cluster B and the encrypted answers are joined together. Finally, a will receive an ordered set $E = \cup E_b$ where each one of the components is the parameter $e(\bar{p}(r), [\bar{q}])$ of the documents that the nodes in B indexed. Next, a decrypts and calculates the distances to the documents, and selects those that are affine according to the threshold λ.

An epidemic algorithm without loops inside the cluster B, as we assumed in sec-

```
Algorithm 2: multi_DB_search_query($b$, $\bar{q}_m$, $depth$, $branches$)
    $S \leftarrow indexer(b)$
    $D \leftarrow \{\}$
    foreach $\bar{p}(r) \in S$ do
        $D \leftarrow D \cup (\sum_{j=0}^{m} [[q_i]]^{p_i})$
    if $depth > 0$ then
        for $i = 0$ to $branches$ do
            $D \leftarrow D \bigcup multi\_DB\_search\_query(b \in_R B, \bar{q}_m, depth - 1, branches)$
    $\{D, \pi\} = permute(D)$
    return $D$
```

```
Algorithm 3: customer_download_pbr($o$, $i$)
    $D \leftarrow \{\}$
    for $b_j \in_E B$ do
        $D \leftarrow D \cup download\_pbr(o_j, [[i]])$
    return $PBR([[i]], S) \cup D$
```

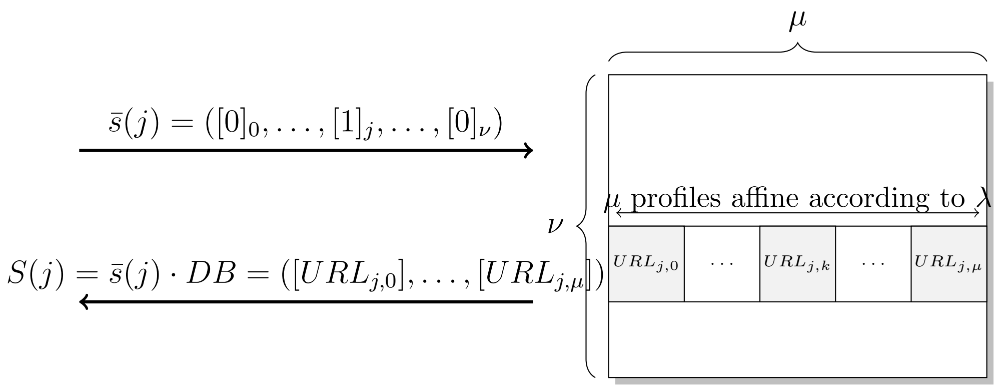
tion 10.2.1, can create the tree of indexers that figure 10.3 shows. For the porpoises of this document, the tree structure has a disadvantage: answers that are provided by nodes in the inner branches of the tree are going to be localized in the last positions of the joint vector E. Since we want to provide indexer anonymity, it is necessary that each one of the nodes locally permutes the vector E, as algorithm 2 shows. This way, the origin of the query cannot identify the position of an indexer in the tree according to the position of its answer within the vector E. In addition, during the GET sub-phase, indexers must locally undo these permutations. The permutation that each indexer apply must be a secret that shouldn't be made public. This mechanism can be refined to avoid duplicated items by means of Bloom's filters.

GET URLs: Private Block Retrieval Protocol

Finally, a private block retrieval (PBR) scheme takes place in order to download the $URL(r)$ associated to the document that is of his interest without noticing b. The private block retrieval scheme hides the index of the item that a is retrieving, and it ensures that nobody in the path $a \rightarrow b$ knows which item a is interested in, not even the indexer b.

For the sake of completeness, we describe next a simple yet useful PBR scheme to get a URL from the indexer without leaking which URL the user is asking for. This PBR system uses the Paillier's crypto-system [93].

In order to improve the efficiency of the communication from the point of view of the number or exchanged bits, we will use a two-dimensional view of the database. Hence, the customer won't receive a single URL, but a set of μ related, affine $URLs$. First, the database needs to be organized in the following way. An indexer creates ν



μ
$\bar{s}(j) = ([0]_0, \ldots, [1]_j, \ldots, [0]_\nu)$
ν
μ profiles affine according to λ
$S(j) = \bar{s}(j) \cdot DB = ([URL_{j,0}], \ldots, [URL_{j,\mu}])$
$URL_{j,0}$
$\cdots$
$URL_{j,k}$
$\cdots$
$URL_{j,\mu}$

Figure 10.4.: The two dimensional PBR system

sets that contain at most μ URLs each, in such a way that the URLs inside a set are affine according to a pre-shared λ, being λ the affinity threshold. These sets are then organized as a matrix of ν rows and μ columns. We assume that the database used an optimization mechanism to calculate a representative of each set, and the distance to this representative profile was sent to the user during the QUERY sub-phase.

Then, the user asks for the j-th row of the matrix, where j was the output of the previous sub-phase. To do this, the customer constructs the selection vector $\bar{s}(j)$ of equation 10.2 and 10.3, where all components are the Paillier's encryption of 0 except the j-th component that stores the encryption of 1. The reader should remember that the Paillier's encryption is probabilistic, and an observer cannot identify the components of this vector.

$$\bar{s}(j) = (s_0, s_1, \ldots, s_k) \tag{10.2}$$

$$\text{where } s_i = \begin{cases} [1] & \text{if } i = j \\ [0] & \text{if } i \neq j \end{cases} \tag{10.3}$$

Then, the indexer multiplies each component of the selection vector for each row of the URL matrix, and adds the results. These operations are shown in equation 10.7, which takes advantage of the homomorphic properties of the Paillier's crypto-system. As figure 10.4 shows, this process results in a vector with the URLs in the i-th row.

$$\begin{aligned} S(j) &= \sum s_i \otimes row_i && (10.4)\\ &= [0] \cdot row_1 + \ldots + [1] \cdot row_j + \ldots [0] \cdot row_\nu && (10.5)\\ &= [row_j] && (10.6)\\ &= ([URL_{j1}], ..., [URL_{j\mu}]) && (10.7) \end{aligned}$$

Further analysis of possible PBR systems can be found in [92]. Even if there are other proposals more efficient than this one, the reader will notice that the PBR system that has been described is very convenient for our proposal, since a user gets a complete row of affine profiles from the indexer with a single query and most of them would be of his interest.

We will clarify the complete recommendation process, including the indexer tree, using an example. Figure 10.5 shows a tree of three indexers that returned an array E with 6 distances. Indexer x contributed with $\{d_1, d_2\}$ to the array, indexer y contributed with $\{d_5, d_6\}$ and indexer z with $\{d_3, d_4\}$. When the user receives the vector of encrypted distances, he chooses the elements 2 and 5, and starts a PBR process. Indexers do not know the position of the elements that a user wants to download, so each x, y and z builds a virtual database that contains the documents that they store in the same position that they answered in E, using empty elements for the other positions. When x receives a petition for d_2 and d_5, it is able to answer d_2 and the $5-th$ position is empty, x_5. z cannot answer any valid document and only returns empty positions, while y answers an empty item and a valid URL d_5. The reader will notice that indexers are unaware whether they are returning valid $URLs$ or just empty items. Then, the user receives the answer array, undoes the PBR scheme and finally discards the empty items.

Using this scheme, it is clear that neither indexers nor an observer in the middle of the path can learn whether a particular document comes or not from a specific indexer. Indexers are unaware even if they gave a valid answer to the user. By means of the additional permutation that was introduced in section 10.2.3, which was not shown in this example in sake of simplicity, the customer cannot either reconstruct the original positions of the indexers within the tree.

10.2.4. DOWNLOAD the document

Finally, the customer will download the desired document from the distributed filesystem. We assume that it is not possible to learn any information of $\bar{p}(d)$ from $URL(d)$ of a document, and that it is not possible to access to the $URL(d)$ from any node not in A without the group key K_A. A secure distributed filesystem that meets these requirements will be studied in part V of this thesis.

10.3. Analysis of DocCloud

This section revisits the system requirements that were introduced in section 9.3.2. These requirements are analyzed according to the security mechanisms and procedures that were described in the last section. In addition, section 10.3.1 explores the indexer anonymity requirement in depth.

Indexer plausible deniability. Indexers store document profiles "projected but in clear". This is not a security risk, and indexers cannot be legally prosecuted due to failure to calculate the original document profile. First, indexers in cluster B won't search actively for the shared key of an external cluster A because they cannot identify it. Even if an attacker informs the indexer about the identity of the cluster A and distributes the

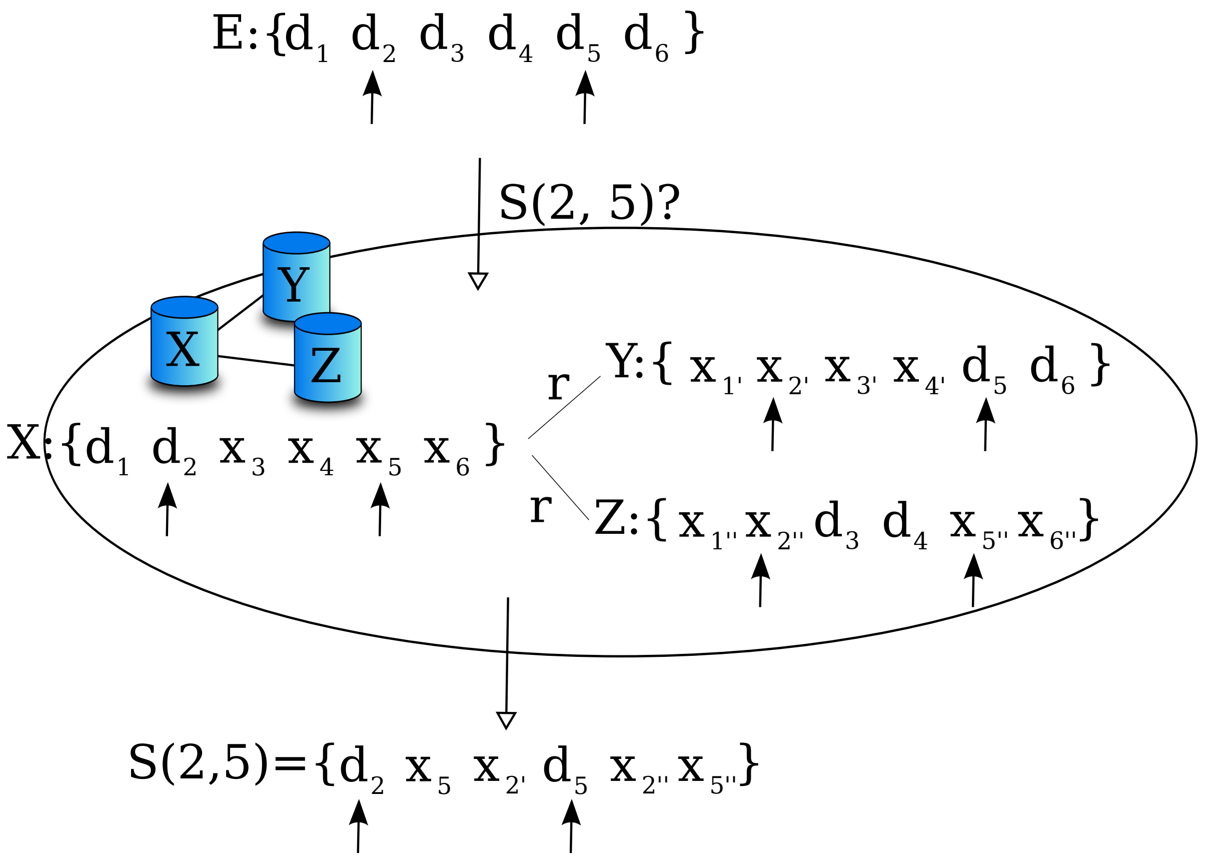

E:{d1 d2 d3 d4 d5 d6 }
S(2, 5)?
Y
X
Z
Y:{ x1' x2' x3' x4' d5 d6 }
r
X:{d1 d2 x3 x4 x5 x6 }
r
Z:{ x1'' x2'' d3 d4 x5'' x6''}
S(2,5)={d2 x5 x2' d5 x2'' x5''}

Figure 10.5.: A PBR scheme in the multi-indexer scenario

shared projection matrix M_A, the projection matrices still hold the undecomposability threshold referred in [82]. Hence, there is not enough components to learn even a single category of the original profile. Second, merchants are hidden in a social network that uses anonymous routing. Hence, indexers are not able to identify the source of a document description and they cannot use any source-analysis mechanism to learn information about the document profiles. Third, since indexers do not distribute documents but only index URLs that are protected with the shared key K_A, they cannot access to the actual document. We conclude that indexers are able to deny that they are able to access to the document that they index, they cannot undo the projection and calculate with enough precision the original profile and they cannot identify the merchant that inserted the profile. This way, the system provides indexer plausible deniability.

Since a malicious user in cluster A is able to make a good guess of the kind of profiles of the other users in the cluster and the document profiles that they share, the attacker is able to inform indexers of the document about the profiles that they store. For this attack to be successful, the malicious user must be able to identify the indexer of a specific document profile. We will study this attack later.

Cloud provider plausible deniability. The cloud provider only observes communication from clients to indexers. Messages during queries use homomorphic encryption and thus they cannot be accessed. The profiles that insertion messages include are projected, and therefore they enjoy the same protection as individual indexers. As observers of the communications and thanks to the PBR system, they cannot identify the specific indexer that provided an answer. Finally, they could access to the virtual machines of the cloud, their memory and data. Even if instances save their databases in clear, they only manage projected profiles. This way, the cloud provider enjoys the same protection as individual databases.

Oblivious routing. Nodes in the middle of the path between a user $a \in A$ and indexers in B route messages between them. They are inside the anonymous cloud, so they are able to learn as much information about the source of a message as the anonymous cloud allows. We propose a system where document profiles are published in a projected social space but in clear. As a consequence, the intermediate nodes are similar to indexers from the security analysis point of view, and therefore the same considerations can be applied to them. However, since profiles will be projected to a dimension $m < \frac{n-1}{2}$, according to [82], the intermediate nodes are not able to calculate any specific component of the profile. During the recommendation phase, messages exchanged by indexers and users are encrypted using a key only known by the client a, and thus intermediate nodes cannot gain any knowledge about messages. We can conclude that intermediate nodes are as protected as indexers during the publication of the profiles, and that they cannot learn anything about queries during the recommendation phase.

Indexer anonymity. The multi PBR scheme that was introduced in section 10.2.3 prevents users from learning the identity of the specific indexer that answered a query. In fact, not even the indexers that take part of the recommendation tree know whether or not they answered the query. Only the user that issued the query is able to distinguish empty items from real information, and in any case the local permutation of the vector of answers in every hop of the tree prevents to gain any knowledge about the indexer identity. This way, the system provides indexer anonymity since not a single entity of the system is able to identify the source of an answer.

The first three requirements are fulfilled using the building blocks described in Section 9.2. Indexer anonymity is the only requirement that does not depend on external mechanisms but depends directly on the structure of the system that we are proposing. Specifically, the property depends on the number of contributions of each indexer that participates in the tree depicted in figure 10.5, as was discussed in section 10.2.3.

In the next section, we will analyze in depth the indexer anonymity property and the management of the indexer tree to maximize this property.

10.3.1. A metric for indexer anonymity

In this section, we will analyze the indexer anonymity property that the system shows, and we will provide a way to calculate the maximum number of items that each indexer must return to provide the plausible deniability to the indexers.

Indexers are organized in a tree as figure 10.5 shows. In section 10.2.3, we learned that items inside the answer vector must be randomly shuffled to prevent the attackers from being able to identify indexers by means of inspection of the positions of the results. Indeed, indexers that are deeper in the tree structure send their profiles to their root, which performs a Bloom's filter to avoid repetition of profiles in the answer. The effect of this filter is that it is more likely that the information of a profile in the answer array comes from the leaves than from the root. In an extreme case, the root of the indexer's tree does not contribute at all to the answer. The attacker is not able to identify the indexer that holds a profile, but he can assign a different probability to each indexer in the tree. In this sense, the anonymity set is biased towards the inner leaves of the tree and it is smaller than expected.

The trivial solution to avoid different contributions is just not to implement Bloom's filters and let that every node contributes to the answer with his entire database. However, we describe an epidemic algorithm to spread document profiles among the indexers of B. Hence, the same document profile is stored by many indexers of B. Not removing duplicates in the answers implies that indexers must recalculate the PBR functions of large vectors, which is a slow process, and the number of exchanged bits grows unnecessarily. We will show that we can still get balanced contributions without the need of sending the whole database during each query.

In this section, we will consider an attacker that learns the list of document's profiles that a given query returns. This may be the case of the sender of a query. We establish that the attacker wants to identify the indexer that stores a specific document profile. To do this, he issues a query that aims exactly to the targeted document profile. We assume that the attacker knows the identity of the indexers that participated in the indexer tree. This is the case, for example, of an attacker acting as a cloud provider, or a malicious indexer inside cluster B.

The main idea of this section is calculating the average size of the answer vector $a = |E|$ in the system of k indexers. Hence, we can force that if the indexer tree has k nodes, then each indexer contributes to E with a/k URLs. In this sense, from the point of view of the client, any document profile could uniformly come from any of the indexers of the tree.

Uniform assumption

Indexers of a cluster B store a set $D = \{d_1, d_2, ..., d_N\}$ of different document profiles. Each indexer stores $n < N$ of them. When a query arrives, the recommender system will randomly pick a subset of indexers $S \subset B$ with cardinality k that contribute to the creation of the answer of the query, as explained in section 10.2.3.

As a first approach, we suppose that document's profiles are uniformly spread in B.

That is to say, given a document d_i, chances that an indexer stores d_i are independent of the indexer. For any indexer $b_u \in B$, we define an event $\bar{X}_{i,u}$ as "the document d_i is not in b_u". As we assume uniform distribution of documents, the probability distribution function (pdf) of X can be modeled as a hypergeometric distribution: given a set of N different documents, $x = 1$ are of our interest. That is, d_i. Then, we pick without replacement n documents and calculate the chances that $k = 0$ of these are d_i.

$$p(\bar{X}_{i,u}) = hypergeom(x = 1; n = n, N = N, k = 0) \tag{10.8}$$

$$= \frac{\binom{1}{0}\binom{N-1}{n}}{\binom{N}{n}} = \frac{N-n}{N} \tag{10.9}$$

Given a subset S of k indexers, each one storing n different document's profiles, we define the event $\bar{Y}_i$ as "the document d_i is not in any of the indexers in S". The complement of this event, Y_i, means that the document d_i is at least in one indexer in S. Since we assume a uniform distribution of documents, the pdf of Y_i is constant for any document and indexer, and from this moment forward we will drop the subscript. Hence, the pdf of $\bar{Y}$ is:

$$pdf(\bar{Y}) = pdf(\bar{X})^k \tag{10.10}$$

$$pdf(Y) = 1 - pdf(\bar{Y}) = 1 - pdf(\bar{X})^k \tag{10.11}$$

Finally, we define an event Z_j as "the subset S has j different documents". Since each indexer stores n different documents, the minimum value of Z_j is n, that is to say, the k indexers are the same. On the other hand, the maximum value of Z_j is nk, and this is the case where the k indexers store completely different documents. Hence, the universe of Z_j is $[n, kn]$. This event is equivalent to "the subset S contents at least one instance of j documents and no instance of $N - j$".

Now, we can calculate the pdf of Z_j as follows.

$$pdf(Z_j) = \begin{cases} \binom{N}{j} pdf(\hat{Y})^{N-j}[1 - pdf(\hat{Y})]^j & \text{if } n \leq j \leq nk \\ 0 & \text{otherwise} \end{cases} \tag{10.12}$$

$$= \begin{cases} \frac{N!}{N^{kN}} \frac{(N-n)^{k(N-j)} n^{ki}}{(N-j)!j!} & \text{if } n \leq j \leq nk \\ 0 & \text{otherwise} \end{cases} \tag{10.13}$$

The pdf of equation 10.13 is a binomial distribution that has been shifted by kn, and therefore its average is:

$$E[Z_j] = (k-1)n \; pdf(X) = \frac{(k-1)n(N-n)^k}{N^k} \tag{10.14}$$

Equation 10.14 captures the expected number of different items in the answered vector. The results of equation 10.14 can be used to improve the anonymity set of the indexers. Indeed, if each of the k indexers contributes with $n = E[Z_j]/k$ items, then the contribution of each indexer to the answer array is likely equal. In order to achieve this, we must encourage that $E[Z_j] = nk$. We call this n the optimal contribution coefficient for the indexers, n_{opt}, since it lets uniform contributions for the indexers and maximizes the anonymity of the set. We can calculate the optimal contribution of each indexer n_{opt} as the n that matches the following condition.

$$E[Z_j] = n_{opt}k = \frac{(k-1)n_{opt}(N-n_{opt})^k}{N^k} \tag{10.15}$$

$$k = \frac{(k-1)(N-n_{opt})^k}{N^k} \tag{10.16}$$

$$n_{opt} = N\left(1 - \sqrt[k]{\frac{k}{k-1}}\right) \tag{10.17}$$

Equation 10.17 shows the optimal contribution of each indexer to achieve maximum anonymity. Alternatively, equation 10.16 shows the optimum number of indexers that must be contacted, for a fixed number of contributions from each indexer.

In the extreme case of $n, k \ll N$, $E[Z_j] \approx kn$ and in order to achieve uniform contributions, each indexer should contribute with $k \approx n$ items, nearly every item in the database. Since there are much more documents in the system than the capacity of an indexer, if the subset of indexers is small (k small), chances of collision are small and indexers can collaborate with every item.

Even if this could simplify the system design, it is not desirable from the point of view of efficiency. Users of the system will want to calculate the affinity to as many profiles as possible to be able to locate the more interesting documents. In this sense, the system will be designed for $kn \approx N$.

Epidemics assumption

In section 10.2.1, we suppose that there is an epidemic routing algorithm in the indexer set. The effect of this algorithm on document's profiles is that it is much more likely for neighbor indexers to share similar document profiles, and the likelihood of replicated data is higher if indexers are adjacent. Hence, in a real system the distribution of profiles is not uniform as we supposed in the last section, and the pdf that equation 10.9 shows will depend on the position of the indexers in the tree. Hence, $pdf(X_j)$ is not a simple hypergeometric distribution as calculated in the simplified scenario. On the contrary, $pdf(X_j)$ must be weighted with the position of the indexer.

As a first approach to analyze this problem, we are going to suppose that indexers are ordered in a line. This is a simplified tree with no branches. As in the last section, the event $\bar{X}_{j,u}$ represents "the document r_j is not in d_u", but this time we define u as the position in line, from the root $u = 0$ to the branch $u = k$. Then, we describe a routing epidemic protocol in such a way that there is two real numbers ϵ and δ such as:

$$P(\bar{X}_{i,u}|\bar{X}_{i,u-1}) = p + \epsilon \tag{10.18}$$
$$P(\bar{X}_{i,u}|X_{i,u-1}) = p - \delta \tag{10.19}$$
$$0 \leq \epsilon + \delta \leq 1 \tag{10.20}$$

being p the probability of the uniform assumption that equation 10.9 shows. These equations may be interpreted as follows: the probability that a document is (is not) in an indexer is higher if it is (is not) in the precedent indexer. Furthermore, we analyze a routing protocol that makes negligible the variation of likelihood of $X_{j,u}$ given $X_{j,v}$.

Equations 10.20 represents a Markov chain of probabilities. It was analyzed for example in [69], and we present next the solution for $P(X_{j,d})$ as a convenience using our notation.

$$P(\hat{X}_{i,u}) = \frac{(p-\delta) - (\epsilon+\delta)^u(\epsilon\delta - (1-p)\delta)}{1-\epsilon-\delta} \tag{10.21}$$

The last term of this equation attenuates with k, and then it is a monodic decreasing function with a maximum of p for $k = 1$. In the new scenario, $P(\bar{X}_{i,u}) \leq p$. The values for ϵ and δ cannot be easily computed since they depend on the actual epidemics algorithm in use, but we can conclude that any epidemics algorithm that we chose should use equation 10.17 as an upper limit for the contribution.

A real scenario with several branches in the indexer's tree is even more complex. The probability $P(X_{j,u})$ follows a Fisher's non-central hypergeometric distribution. In order to calculate the new pdfs or achieve similar conclusions to the last section, we need to model the epidemics algorithm that the indexer set uses. The specific k_{opt} that maximizes anonymity depends on the details of the epidemic algorithm that is used in the social network.

Discussion

Even if we do not achieve a final result for the probability, we can extract some conclusions from the last scenario. The Fisher's non-central hypergeometric distribution is always shifted toward the left and its average is less than the average of the central hypergeometric distribution. Besides, the analysis of this section for a simplified tree showed that the Markov chain that epidemic algorithm creates always decreases $P(X_{j,u})$. In practice, this means that we can use Eq. 10.17 as an upper limit for the amount of collaboration of the indexers of the system.

Figure 10.6 clarifies the analysis of this section. We used a recommender network that indexes $N = 1000$ documents under the uniform assumption. The figure on the left represents the expected size of the answered vector for different sizes of the indexer tree. For a tree of $k = 10$ indexers, if each indexer contributes with $n = 80$ elements the answered vector has an expected length of 300 elements. The right side of the figure shows the anonymity loss of the first indexer inside the tree. In the last example ($k = 10$, $n = 80$), the indexer in the inner leaves of the tree was the source of an item 0.2 of the

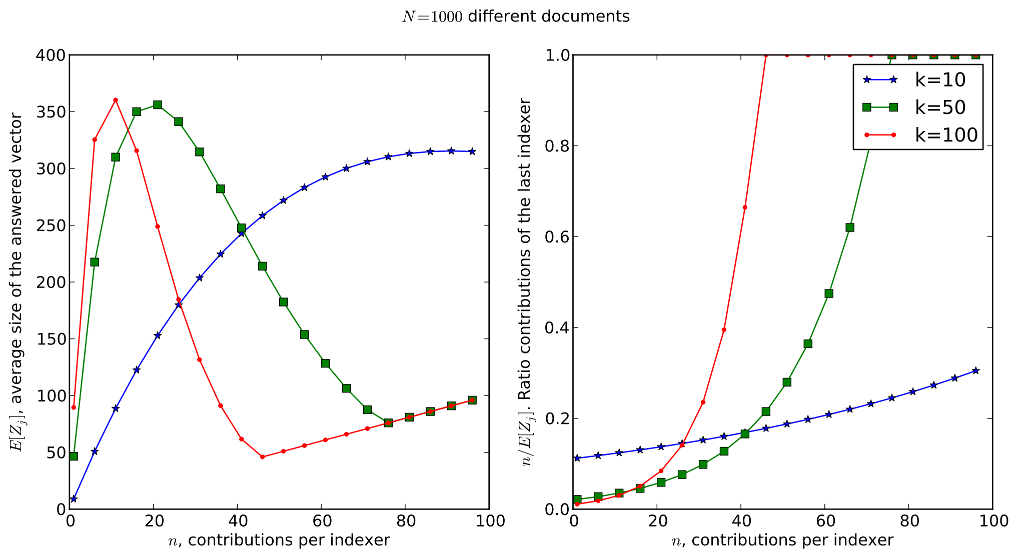


Figure 10.6.: Analysis of contributions for $N = 1000$ documents

time. Since the anonymity set of the tree has a size of $k = 10$ elements, attackers learn that the inner indexers are the source of an item about twice the time than in the maximum anonymity scenario.

For $k = 50$, the scenario is similar: the maximum length of the answered vector of affine document profiles occurs when each node contributes with $n = 20$ items. In this case, the inner indexer is the source of an item with probability 0.05, when the maximum anonymity occurs at $1/k = 0.02$. In this case, the probability of an item to come from an indexer doubles the maximum anonymity scenario, and inner indexers in a tree of $k = 50$ indexers when each contribute with $n = 20$ items, are as protected as if the tree has only $k' = 25$ indexers.

These figures and the equations from this section can be used to decide the number of contributions from each indexer, the apparent size of the anonymity set and the expected size of the returned vector. Larger returned vectors enhance the efficiency of the system, since more affine documents are discovered during a query. But if the number of indexers in the indexer tree is not chosen accordingly, the anonymity loss of the indexers that first contribute to the answered vector may be unacceptably high.

11. Conclusions

In this part of the document, we described DocCloud, a system that protects machines and providers of a document recommendation system from legal attacks.

We established that *plausible deniability* was a necessary security service that recommender system must provide, since it protects all participants of the system against legal attacks. We defined plausible deniability as the ability that a node has to deny any knowledge of the document that they are recommending to the users of the system. Absolute deniability is not feasible in our system, and we use a probabilistic approach to provide this service.

During the development of DocCloud, we made use of the results of previous parts. The users' privacy is protected with the mechanisms proposed in part II of the thesis. Profiles are distorted using a group key that is not directly available to indexers, but indexers are still able to calculate affinities between the different profiles. In the event that indexers learn this group key, the distortion cannot be undone. Furthermore, the publishers of the profiles into the indexers are behind an anonymous network and cannot be identified.

In addition, a Private Block Retrieval scheme that connects customers and recommenders was defined. This scheme ensures that recommenders cannot identify the profile of the document that they are providing to the user. This is not only a safeguard that protects the user's privacy, but also prevents that recommenders can be prosecuted by aiding in in the process of downloading a protected document and allows intermediate nodes the security service of oblivious routing.

Finally, we proposed an organization of databases in a tree-shaped structure to prevent the identification of the source of the recommendation, and provided plausible deniability to databases. Furthermore, this tree structure lets the customer to download an item from the database without leaking the identity of the database that answered the query. Even the database was not able that it was answering a specific query. We explored two different assumptions for the distribution of document profiles inside the tree structure: a uniform distribution and a social distribution. The former is easier to analyze, but the latter is more similar to the organization of nodes in our recommender system. Finally, we provided an upper limit on the number of items that indexers must answer in order to provide optimal deniability inside the indexers tree.

Part IV.

Fast clustering of users

Part IV: Fast clustering of users

The success and intensive use of social networks makes strategies for efficient document location a hot topic of research. In this part of the thesis, we propose a common vector space to describe documents and users to create a social network based on affinities, and explore epidemic routing to recommend documents according to the user's interests. Furthermore, we propose the creation of a SoftDHT structure to improve the recommendation results. Using these mechanisms, an efficient document recommender system with a fast organization of clusters of users based on their affinity can be provided, preventing the creation of unlinked communities. We show through simulations that the proposed system has a short convergence time and exhibits a high recall ratio.

This part of the thesis is structured as follows. Chapter 12 introduces the related work in Section 12.1 on document location and recommender systems; Section 12.2.1 defines the scenario and the common description model for users, documents and queries; and finally, the objective of this part is formally defined in Section 12.3. Chapter 13 includes the protocols to create the clustered network, Section 13.1 proposes and analyzes an epidemic algorithm for the quick creation of the social structure, as well as additional mechanisms to enhance the efficiency of searches; these proposals are simulated in Section 13.4 to test the efficiency of the algorithms.

12. Searching resources in the system

Current P2P networks tend to be huge, managing millions of documents that are distributed among hundreds of thousands of nodes. Finding a specific document in this cyclopean library of unspeakable geometry may be a task doable only by the UU's Librarian. Currently, users typically perform searches via keyword matching: they send a list of keywords to the network and get a set of documents with descriptions containing some or all of the keywords from the list. In order to use these lists, users must know in advance what they want to look for, and discovering new and interesting content is a hard task.

Social networks are a kind of P2P network where users create links according to their similarities. Indeed, the fact that a link exists in a social network means that two nodes conclude that they share some common features. We can take advantage of an existing social network to provide a recommender system similar to the one described by Yager in [146]. In a recommender system, instead of posting a keyword list, users share their likes and dislikes, and ask the community for tips on documents that may be of interest to them. Two users that share a link in a social network probably also have common interests and similar likes. Hence, documents owned by one of them will likely be of some interest to the other. In this regard, a recommender system may use the neighbors of a user in a social network to improve its recommendations. In a social network where users link with similar friends, the problem of searching for documents boils down to asking for recommendations from a small neighborhood.

In this chapter, we aim to improve the performance of our recommender system based on the creation of a social overlay on top of an unstructured P2P network. Users are clustered according to their affinity, and after that they can recommend documents to their neighbors. For this approach to be successful, it requires fast identification and location of clusters or other users that are similar, and an efficient construction of these clusters. In building the social network, we described a common way of describing documents, users and queries. This model was described in chapter 5. These descriptions will make possible to not only compare queries and documents, but also users. That would enable the social network to take advantage of the mechanisms proposed in this part for finding users that are alike and creating links according to the users' affinities. On this social network, our document recommender system is easily deployable.

12.1. Related work

Epidemic searches have often been proposed for routing messages in recommender systems. Furthermore, the definition and classification of documents in a social space is an open field of research. In this section we describe and explore some related proposals in the literature.

12.1.1. Epidemic searches

In a P2P network, epidemic algorithms seem to be a natural solution for searches. Eguster et al. [44] is a good introduction to the main aspects of epidemic algorithms. This kind of routing is often compared to flooding, however information is not disseminated to every neighbor of a node but only to a small subset of them. Hopefully, the results in terms of reachness are close to flooding, but epidemic algorithms are far more efficient with regard network use. Drost et al. [40] explore the problems that epidemic algorithms must face: parallel searches, trust and limited connectivity.

BuddyCast [100] used a simple algorithm for searches based on random walks through friends and friends of friends. Two drawbacks of this protocol are that (i) it does not take into account the links that friends share with each other, and (ii) the creation of the initial links is entirely manual. Even then, according to its authors, BuddyCast is capable of handling hundreds of simultaneous users [99].

Anglade et al. [14] evaluated the previous approach and introduced the comparison parameter that is shown in Equation 2.7, named the overload fraction f_o. This parameter represents the percentage of interesting documents in the network that an algorithm is able to find. We can formally define this fraction in this way. With Q being the set of total queries in the network, $R_{Epidemics}(q)$ represents the set of answers to a query $q \in Q$ that a user obtained using an epidemic algorithm, and $R_{Exact}(q)$ represents the actual number of documents in the network that match the query.

$$f_o = \frac{\sum_{q \in Q} |R_{Epidemics}(q) \cap R_{Exact}(q)|}{\sum_{q \in Q} |R_{Exact}(q)|} \tag{12.1}$$

In the Information Retrieval field, this parameter is equivalent to the **recall ratio** [17]. The recall is the percentage of the interesting documents that are retrieved. In the general case, it is possible to retrieve documents that are not interesting (false positives). In section 12.2.1 we will model "interestingness" in an unambiguous way. Consequently, it will not be possible to retrieve uninteresting documents. Thus, the recall ratio as defined in the Information Retrieval field and the overload function as described in Equation 2.7 are equivalent.

Additionally, Anglade et al. [14] proposed a methodology to evaluate clusters of users with shared interests. However, these clusters must be defined prior to the creation of the network and in that case, the number of possible clusters is static, predetermined and limited. Anglade et al. proposed the metric defined by Resnick et al. [106] to calculate affinities between documents. However, if two users wish to calculate the affinity between them, they must have a certain number of documents in common. If the number of documents that both users have in common is small, the calculated affinity may make little sense. If they have no common documents, the affinity cannot be computed. Hence, joining the social network is a painful process where users must evaluate hundreds of documents and identify their friends before any search can be performed. In the literature, this is described as the *sparsity problem* [143]. Even in this case, since it is very likely that the user will always interact with the same set of neighbors, there is a high likelihood of having closed clusters with the same interests that do not see each other.

Schifanella et al. [113] used epidemic algorithms to distribute the assessment that users make about some documents in the system, and using this information the system can dynamically create a list of similar users. The algorithm was specially developed for ad-hoc networks and only to distribute document assessments, so it exhibits the same problem cited above.

Ruffo et al. [110] took advantage of the small-world behavior that most social networks exhibit. This system calculated affinity based on the number of common documents that two users have, and so if two users have no documents in common, the affinity between them cannot be calculated.

SENSE [31] is a recommender system in a decentralized network with a precomputed global database. It classifies documents in three different groups: *socials* are those inside the cluster of friends explicitly named by the user; *spirituals* are documents recommended by other users that are like the searcher; and *globals* are documents recommended by every node in the network, regardless of their affinity. Searches and affinity are calculated from the overlap of keywords in the common documents.

12.1.2. Vectors as profiles

We follow the social model that was introduced in chapter 5. As established previously, documents are usually modeled as vectors in the Information Retrieval field [85, 124]. In many cases, the components of these vectors are the frequency of appearance of certain common terms in the document under analysis. This same idea can be generalized to include ontologies or categories of terms. Indeed, it is possible to convert a vector of terms (*bag of words*) into a vector of ontologies or categories (*bag of concepts*). The calculation of the ontology that classifies documents falls beyond the scope of this thesis. If interested, readers may refer recent works in the fields of Information Retrieval and Artificial Intelligence [85, 125, 86, 129].

One of the main metrics used to calculate the affinity of two documents is the cosine metric, introduced in chapter 5. Yee et al. [147] simulated the cosine metric in a P2P network, and it is shown that it may enhance search results [106, 85, 79]. The task of choosing a specific metric to compare how close two profiles are also falls beyond the scope of this document. From this point forward, we assume the existence of an affinity metric for profiles. If interested, readers may refer to [13, 87].

12.1.3. Recommendation systems

In this part of the thesis, users query a decentralized, unstructured P2P network to find the documents that they would be most interested in. This is the definition of a recommender system, as we show in partI. Customers are not looking for a specific document, but request recommendations from other nodes in the network. This is the same concept as in [14], but we will use the vector descriptions of [147] which will allow us to use the same mathematical tools to compare documents as well as users. In this regard, we can construct a social network similar to the one described in [31], but in a decentralized fashion. Authors are convinced that this enhances the efficiency of the searches, allows for a more dynamic behavior of users and facilitates the introduction of security mechanisms to protect them.

Many different recommender systems have been proposed, based on either a P2P architecture [14, 113, 100] or the use of a cloud [79]. Recommender systems are characterized by the fact they must handle large amounts of data that users wish to filter according to their profiles. In a recommender system, a user removes irrelevant data by classifying documents according to the opinion that similar users share about documents. In this regard, the problems that a recommender system should face are the creation of the initial social network where users that are similar meet each other, and how to distribute queries in the social network to maximize the efficiency of searches. In this thesis, we focus on a collaborative recommender system. In this type of system, the filtering of documents depends on the opinion that similar users have about a document. In this regard, the social network should be constructed according to the affinity of users.

Fuzzy clusters, where users are organized in clusters based on their affinities, are used in some recent recommender systems. Teran et al. [124] proposes fuzzy clustering for eElections. The number of clusters is closed, and a gradual membership function is used. There is no need of a membership function in our recommender system: fuzzy cluster are used only during the creation and maintenance of the social network, and recommendations do not depend directly on the membership value. Gong et al. [51] proposes a clustered network to solve the problem of the sparsity of resources in recommender systems. However, the process of creation of the similarity clusters is not decentralized, and a central entity that is aware of all users and their profiles must be provided. Furthermore, authors only use one rule to select neighbors, which we will refer as the “similarity criterion”. In this work, we will explore some additional rules as well as other mechanisms to improve the recall ratio of the recommendations. Xhoe et al. [152] proposed a hybrid recommender system that uses both “accuracy” (similarity) and “diversity” (success) rules. They conclude that many recommendations do not come from close friends, but from people with a limited connection (“weak ties”). Their system cannot be directly mapped onto a distributed P2P network, but we will use their conclusions to develop our system.

12.1.4. DHT for Recommendation Systems

In our work, we will introduce for the first time the concept of a SoftDHT to enhance the results that the recommender system outputs. As we will see, our SoftDHT will allow the creation of the social structure much faster than traditional protocols.

Authors in [13] proposed a structured P2P called *FuzzyDHT*. Despite having a similar name, the proposal of Andreolini et al. is completely different from our SoftDHT. The Fuzzy DHT described in [13] allows for searching documents according to a list of

Table 12.1.: Notation used in this part of the document

N	Number of nodes in the network
k	Number of outgoing links of a node. It is a parameter of the network, and it is constant.
Neighborhood of a node u, Γ_u^1	Set of nodes that u links to
nth-neighborhood of node u, Γ_u^n	Set of nodes that nodes in Γ_u^{n-1} link to and that are not included in lower neighborhoods. Nodes of Γ_u^n are at n hops from u, and no less
nth-neighbor distribution of a node u, Λ_u^n	$\Lambda_u^n = \cup_{i=0}^n \Gamma_u^i$. Set of nodes that are reachable with n hops or fewer from u.
Recall ratio	Fraction of the documents relevant to the query that are successfully retrieved. In the scenario of our study, there are no false positives and thus the recall is the ratio between the interesting documents found and existing.
Clustering coefficient, $\gamma_u(v)$	Ratio of neighbors that u and v share. γ_u is the average of the clustering coefficient for all neighbors of u, and γ the average of γ_u for all users.

keywords. In this thesis, we aim to provide a recommender system that is completely based on the affinity with our neighbors. As such, we will not use our SoftDHT to handle keywords and find documents, but instead users whose interests are similar to ours.

12.2. Models and definitions

In this section, we define the notation and the assumptions that we will make in this part of the document. Table 12.1 summarizes the main parameters that we use during this work, as a convenience to the reader. Detailed descriptions of these parameters and additional names are introduced in the rest of the section.

12.2.1. Social space and profiles

Documents, users and queries are modeled using vectors the social space introduced in chapter 5. As described in chapter 5, it is possible to define n semantic categories to classify all documents in the system. Then, each document $r_i \in R$ can be associated with a vector within this ontology $\bar{p}(r_i) = \{c_1, c_2, \ldots, c_n\}$. We call this vector the **document profile**. The space of all possible profiles is the social space of our system. As established in section 5.2, for the sake of simplicity we model our social space with a constant number of categories, and hence profiles are vectors of a fixed size. With a more complex definition of the metric function that is used to calculate affinities, the routing protocol that is proposed in this part of the thesis may be applied to a social model with vectors of a variable size.

These similarity functions used to compare vectors and calculate their affinities were discussed in section 5.2. In this part of the thesis, we use the similarity based on the cosine metric that was presented in Equation 5.8. This metric was successfully used in the past to compare document profiles [106, 85, 79].

As a reminder to the reader, in Equation 5.8, $s(\bar{p}_1, \bar{p}_2) = 0$ shows that profiles $\bar{p}_1$ and $\bar{p}_2$ are completely dissimilar, and $s(\bar{p}_1, \bar{p}_2) = 1$ shows that $\bar{p}_1$ and $\bar{p}_2$ are completely similar.

In chapter 5, we defined that two profiles $\bar{p}$ and $\bar{q}$ are affine if $aff(\bar{p}, \bar{q}, \lambda) = True$. If $\bar{p}_u$ is the profile of a user u, $\bar{p}_d$ is the profile of a document d, and these profiles are affine, we say that d is an **interesting document** for u. We call this metric the **affinity of two profiles**.

Hence, documents, users and queries may be defined as vectors of $\mathbb{P}$ and there is an unambiguous metric to calculate the affinity between them.

12.2.2. Network structure

In our system, users are organized in a social network that is mapped onto a P2P network. We can consider this network as a symmetric graph $G = \{V, L\}$, where V is the set of nodes in the network and L the set of links between them. We call the neighborhood of a node u, Γ_u^1, to the set of nodes in V that has a link in L between them and u. By induction, we use the term Γ_u^i to refer to the set of nodes in V that has a link in L to a node in Γ_u^{i-1} and it is not present in any lower neighborhood. We use the term distribution Λ_u^i to refer to the set that results in a union of every neighbor $j \leq i$, $\Lambda_u^i = \cup_{j \leq i} \Gamma_u^j$. We use the term *degree of a node* k_u to refer to the number of links that it has, that is $k_u = |\Gamma_u^1|$. In our scenario, we define a network where the degree of every node is constant and much less than N, $(\forall\ u \in V,\ k_u = k \ll N)$. The diameter of the network is the maximum distance in network hops between two nodes. We define the clustering coefficient of node u as in [145], using Equation 12.2:

$$\gamma_u = \frac{|E(\Gamma_u^1)|}{\binom{k}{2}}, \tag{12.2}$$

where $|E(\Gamma_u^1)|$ is the number of edges in the neighborhood of u, $\binom{k}{2}$ is the number of possible edges and γ_u captures the percentage of shared neighbors that u and his neighbor have. In any highly clustered network, the Γ^1 of two linked nodes is nearly the same. The aggregated clustering coefficient, γ, is the average of all the relative clustering coefficients.

Liu et al. [83] demonstrated how social networks show small-world behavior [145]. Small-world networks have low diameter and a high clustering coefficient. The assumption that P2P networks exhibit small-world behavior has been successfully used for document location and routing in [89, 99, 81, 56].

Research in networks with small-world behavior shows an interesting property that Watts et al. named “the strongness of weak links” [145]. In his book, Watts stated that a route between any two nodes will very likely pass through certain special nodes, called *shortcut nodes*. These nodes exhibit a very low clustering coefficient γ. If we define that two nodes with a high γ are part of the same community, these shortcut nodes are not actually members of any community. This does not mean that they have a low number of outgoing links, but rather that nodes they link to are not linked to each other. Figure 12.1 shows an example of a small-world network with shortcut nodes.

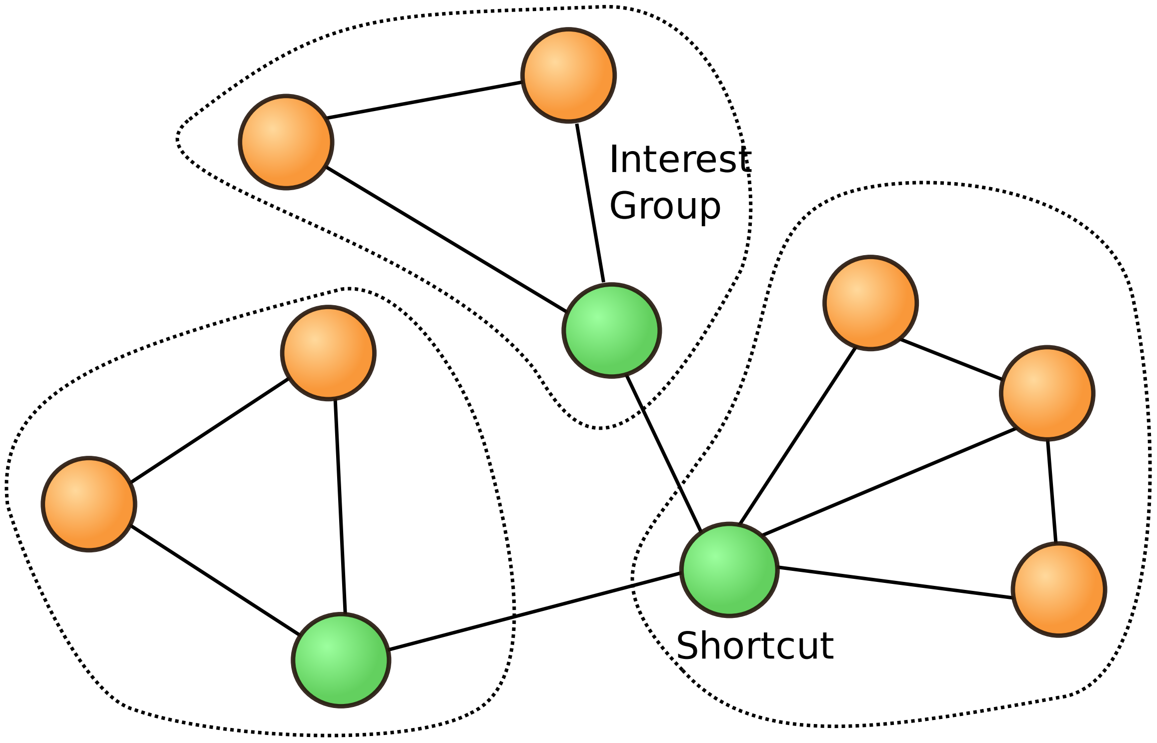


Figure 12.1.: Interest groups and shortcut nodes

We will show that nodes with low γ are shortcuts between clusters that do not know each other, but that may have shared interests. Indeed, tastes are transitive: if two different nodes decided to link to the same shortcut, there is a high probably that they are alike and should meet each other.

In this thesis, we make the additional assumption that it is possible to calculate the γ of a node to any of its neighbors. A trivial solution is that every node shares the Γ^1 set with its neighbors, and thus all of them are able to calculate their γ.

12.3. Scenario and objectives

Using the models and definitions from the last section, we can now define the scenario and objective of our thesis.

12.3.1. Scenario

We devise a system where users of a million-size network organize in large groups of users that roughly share some common ground, and then fine-tune their searches inside these huge groups using the mechanisms that are provided in this document. For example, let's imagine a social network like Facebook with a hundred million users. Within Facebook, it is possible to create and join specific groups such as "I like movies." Through these groups, an initial social network of millions is now structured in several clusters of thousands of users that are at least roughly similar. Inside these large groups, it is possible to get enhanced document recommendations by additional clusterization using our proposed mechanisms. Even general groups such as "nationality" or "age" will work. The same digression works, for example, in a professional network like Akademia: there are millions of potential users, but it is possible to group them by areas of interest (computing science, physics, biology, etc) with only a few thousand and then apply automatic clustering to fine-tuned searches.

12.3.2. Phases of the recommender system

In the social network that we are studying in this part of the thesis, we will simplify the process of recovering recommendations to these phases.

Creation of a connection structure . The objective is to induce the characteristics of a small-world in the network, where nodes link to each other mainly according to the affinity of their descriptions. In [145, 29], it is proved that a few random connections in a highly clustered network may induce this small-world behavior.

Dissemination of information about the documents shared by a user. In our scenario, we disseminate the pair $(\bar{p}(r), URL(r))$ that is associated with a document $r \in R$. This way, documents can be found and downloaded even if the node that initially introduced them in the network is no longer online. This scheme is similar to the one used in [27, 73].

Document location in the network based on affinities. A user shows interest in the description of a document, and searches the network for documents that match the description.

Publication and download of documents . The results of the previous phase are URLs to download a document, and in this phase we take advantage of the P2P network mechanisms to download the document.

12.3.3. Objective

The objective of this part of the thesis is that given a query $\bar{q}$ by a user u and an affinity function (based on threshold λ), it is possible to efficiently find documents in the network that are affine to $\bar{q}$. That is to say, to retrieve as fast as possible the maximum number of elements of R_q:

$$R_q = \{r \in R \mid aff(r, q) = True\} \tag{12.3}$$

Therefore, in this part we focus on phases one and three, using the creation of a suitable structure for searches. Part III of this thesis explored phase two. We separate the problem of document location from that of document download due to security reasons. The phase four of this simplified scenario uses the mechanisms that will be described in part V.

13. Fast clustering of users

P2P networks that are structured as social networks can use the small-world behavior to enhance the results of the search mechanisms and offer recommendations to the users. In these networks, nodes in Γ_u^1 are related to the interests of a user u, and they have additional random links to maintain the structure of a small world. Hence, it is extremely likely that nodes in Γ_u^1 share the same interests than u.

Several proposals of epidemic algorithms for this kind of networks were explored in section 12.1.1. We will create open and distributed groups or **clusters** of users that are linked according to their affinity. Each node will have its own view of the group, and therefore it has no well-defined border.

In our design, links are dynamic. Hence, users do not need to share or evaluate long lists of documents to calculate their affinities. Unlike the proposals described in Section 12.1.1, users join the network using their user's profile based on the profiles of the documents that they shared, as it was introduced in chapter 5.

In this chapter, we explore first a simple proposal to create clusters. Then, we perform some initial evaluations of this preliminary proposal and propose new mechanisms to improve the recall parameter of our recommender system.

13.1. Basic clustering of users

13.1.1. Joining the network

When a new node v joins the network, it contacts to a random node u that was previously inside the network and requests to "search users that are similar to me (v)". Then, v searches their own profile by means of querying nodes in Γ_v^1 (Algorithm 4).

Algorithm 4: join_network(λ), run by a new user v

```
Data: New user of the network, $v$
Input: $\lambda$, the similarity threshold for the affinity function
Output: $v$ links to some nodes of the network
begin
    foreach $u \in \Gamma_v^1$ do
        S ⟵ {$u$:search_user($v$, $\lambda$, 1)}
        foreach $w \in S$ do
            add user $w$ to $\Gamma_v^1$ (optionally, make room)
```

In general, we assume that a new node v enters the network with the ability of finding random nodes that are already members of the network. Actually, real implementations of social networks rarely start with random friends. Often, new users link initially to those who are their friends in real life, and it is highly likely that they share some interests. Linking to users that are initially affine, as real social networks do, would greatly improve the performance of recommendations during the initial searches. In this research, we want to test the merits of the protocol when a user starts from a truly random situation, and therefore we will consider the worst case of $\Gamma_v^1 \subset_R V$ being random during the initial phase.

If v gets an answer from an similar node, it is joined to Γ_v^1. We set a limit to the number of elements in Γ_v^1: if a new affine node is found and there is no room in the neighborhood, the less-similar node is removed to make room for the new one.

Algorithm 5: search_user(v, λ, hops). Run by u, original caller is v

```
Data: Current user $u$, and her attribute $\bar{p}_u$
Input: Calling user $v$, $\lambda$, current $hops$
Output: Set $S$ of users such as $\forall\ s \in S,\ aff(\bar{p}_s, \bar{p}_u, \lambda) = True$
begin
    $S \longleftarrow \emptyset$
    if $aff(\bar{p}_u, \bar{p}_v, \lambda)$ then
        append $u$ to $S$
    if $hops \leq H$ then
        foreach $w \in$ select_next_users($v$) do
            add all elements of $\{w$: search_user($v$, $\lambda$, $hops + 1$)$\}$ to $S$
    return $S$
```

Algorithm 5 shows the code run by nodes u that are already in the network. First, nodes check whether they are affine with the initial node or not. Next, they check if the maximum number of hops has been reached. If not, they forward the query to a selected subset of their Γ_u^1, and the process goes on until the maximum number of hops is reached.

Algorithm 6 shows how the subset of Γ_u^1 is selected.

Algorithm 6: select_next_user(v)

```
Data: Current user $u$, and her attributes $\Gamma_u^1$
Input: Caller user $v$
Output: Set $R$ of users to forward the query
begin
    $R \longleftarrow \emptyset$
    $S \longleftarrow$ order_by_similarity($\Gamma_u^1$, $\bar{p}_v$)
    append to $R$ the $M_{sim}$ most similar elements of $S$
    $S \longleftarrow$ order_by_clustering($\Gamma_u^1$)
    append to $R$ the $M_{clus}$ less clustered elements of $S$
    append to $R$ $M_{rand}$ random nodes of $\Gamma_u^1$
    return $R$
```

We are defining an epidemic algorithm where nodes will choose a subset of Γ_v^2 to forward messages using three parameters: M_{sim} more-similar nodes, M_{clus} less-clustered nodes and M_{rand} random nodes. The process goes on until a predefined number of hops H is reached.

All three criteria or rules of routing, M_{sim}, M_{clus} and M_{rand}, are important for the epidemics of the algorithm. First, the similarity criterion through parameter M_{sim} enables finding users that are affine in the social space. It is easier to find interesting documents inside affine nodes, and this is the same mechanism that was used in [14, 100]. Meanwhile, nodes that are alike in the social space are likely linked to each other, and thus they have a high relative γ. Therefore, through parameter M_{clus}, the node sends the query to nodes that are **not** in the same cluster as the user, disregarding their similarity. This rule has been less explored in the literature. It was used for group creation in [14], but not for routing searches. The intuition that supports the use of M_{clus} is that it enables to arrive to nodes of the social space that share some interests but do not yet belong to the same cluster as the user. Nodes with a low clustering coefficient are, therefore, shortcuts to other clusters in the network. Finally, the criterion of random nodes through parameter M_{rand} is used in many random-walk and epidemic protocols, and it will be evaluated in our research as well. It is the main rule during the first steps of a new user in the network, since they don't know anybody affine and will try random links at first.

After a round of these algorithms, v will have in Γ_v^1 either nodes that are alike to them, or the same initial Γ_v^1. In the simulation section, we will test how good this algorithm is at finding similar nodes during the initial phase.

13.1.2. Browsing the network

After the joining phase, a user v should run Algorithm 4 periodically in order to find new nodes that are affine to them. Networks are dynamic and links are created and destroyed all the time. If a user is not able to find another user that is affine at a given moment, it may be possible later on. In this regard, there is a "slow joining period" before the maximum performance in the network is reached. During this period, users are learning about the structure of the social network.

Proposals presented in Section 12.1.1 do not have any mechanism to find users that are affine, since they only search documents. We described users and documents as vectors in the same social space, and we use a common metric to calculate affinity between users, documents and queries. Thus, mechanisms to find affine users can be provided, improving the quality of the document recommendations.

After joining the network, users will try to locate documents. The process is started by a node v, which selects a query $\bar{q}$ and a threshold λ, and runs algorithm 7 over their own Γ^1_v.

Algorithm 7: get_recommendation($\bar{q}$, v, λ, hops)

```
Data: Current node $u$, her set $\Gamma^1_u$, the set of shared documents $R_u$
Input: Query $\bar{q}$, caller node $v$, $\lambda$, number of current hops
Output: Set $R$ of recommended documents
begin
    $R \longleftarrow \emptyset$
    foreach $r \in R_u$ do
        if $aff(\bar{q}, \bar{p}_r, \lambda)$ then
            Append $r$ to $R$
            if $v \notin \Gamma^1_u$ then
                $u$:user_found($v$)
    if $hops \leq H$ then
        foreach $n \in select\_next(\bar{q}, \Gamma^1_u)$ do
            Append {get_recommendation($\bar{q}$, $v$, hops+1)} to $R$
    return $R$
```

Algorithm 7 is similar to Algorithm 5, but looking for documents instead of users. A mechanism to create new links is introduced in Algorithm 7: when a user u receives a query $\bar{q}$ from a user v and they have at least one document that matches the query, they test whether or not v is in Γ^1_u. If they are not, then u links to v. That is to say, if a previously unknown user v is interested in one of the documents of u, then u assumes that v is affine and tries to create a link in Γ^1_u.

Algorithm 8: select_next($\bar{q}$)

```
Data: Current user $u$, and her attributes $\Gamma^1_u$
Input: The query $\bar{q}$
Output: Set $R$ of users to forward the query
begin
    $R \longleftarrow \emptyset$
    $S \longleftarrow$ order_by_similarity($\Gamma^1_u$, $\bar{q}$)
    Append $M_{sim}$ most similar nodes of $S$ to $R$
    s $\leftarrow$ order_by_clustering($\Gamma^1_u$)
    Append $M_{clus}$ less clustered nodes of $S$ to $R$
    Append $M_{rand}$ random nodes of $\Gamma^1_u$ to $R$
    return $R$
```

The main difference between Algorithms 6 and 8 is that the last one looks for documents instead of users, but readers will notice it uses the same parameters M_{rand}, M_{sim} and M_{clus} that were defined in Section 8.

The algorithm of directed searches that has been presented in this section helps to limit the use of the network without reducing the recall ratio. We will demonstrate this behavior in Section 13.4.

13.2. Performance evaluation of basic clustering of users

The first tests performed on the previously proposed protocol showed that it created many groups of users. Some of these clusters were redundant. These were disjoint groups of users that share interests but do not link to each other. As a result, the first version of the protocol led to a low recall ratio. Running the system as described, once a user u locates and links to a user v with similar interests, u will likely discover and link to v's neighbors. Hence, it is difficult for u to locate new interesting people that are not part of v's group and the current proposal creates an inefficient "endogamy" inside clusters. Actually, users within different clusters that have the same interests will eventually locate each other, thanks to the learning capabilities of the system, but the process is unacceptably slow. We discuss these findings next.

Figure 13.1 shows the recall ratio depending on the network size. Since the users' knowledge about the network structure increases using their own searches and the searches that they route, the recall ratio intuitively increases over time. This intuition will be tested in our final simulations, but in this section, we simulate only a single search from every user in the network to analyze the convergence time of the proposed algorithms. The recall ratio of a flooding algorithm with a limited number of hops ($H = 5$) is also presented. As the figure shows, the proposed algorithm performs worse than a simple flooding after only a single search.

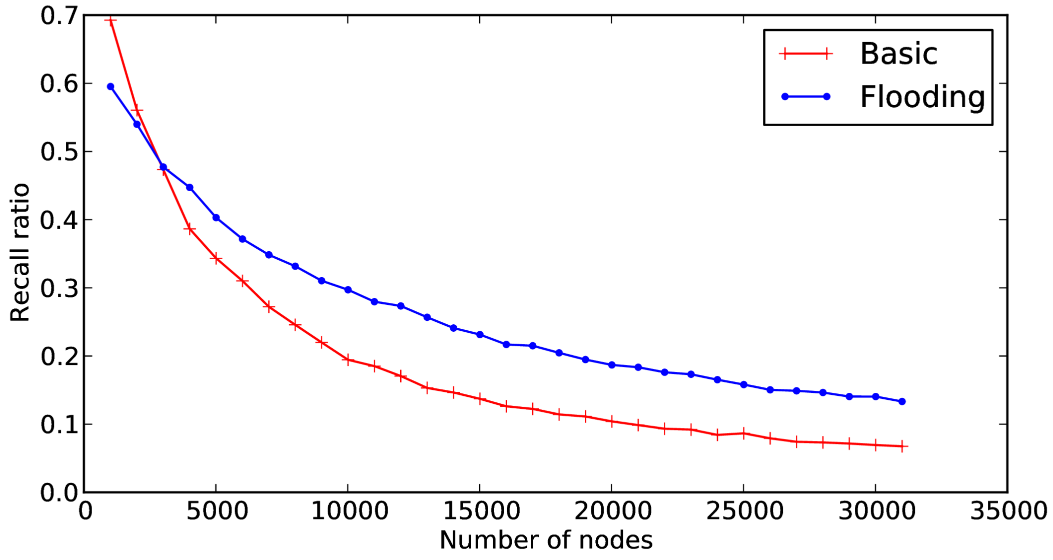


Figure 13.1.: Recall for different network sizes and links. Only one search per user

Figure 13.2 gives a hint about the cause of the problem. It shows the average number of hops in the network that are needed to get an interesting recommendation [1]. This

[1]In Section 13.4.3 we will show that in networks with $N \leq 5\ 000$, 5 hops are probably enough to reach all nodes in the network. Therefore, a recall ratio of nearly 1 for the flooding algorithm is consistent, since most nodes of the network are reached.

figure shows that our simple proposal gets its recommendations mostly at 2-3 hops (that is, from nodes in Λ^3). This suggests that there are some groups of users with the same interests and at least 4 hops apart from each other that are not able to link and share recommendations. These are the “constellations” of our network.

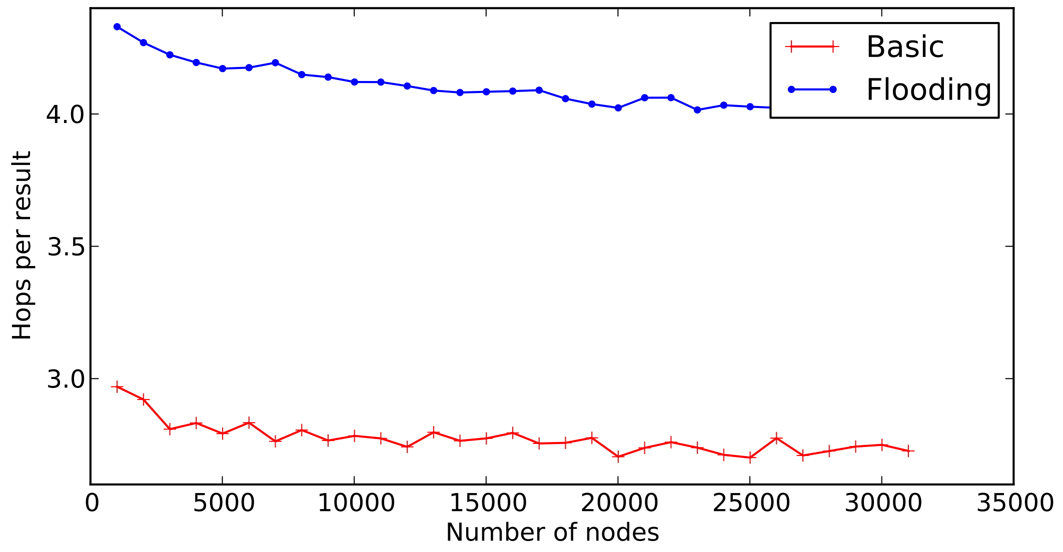


Figure 13.2.: Average hops per search. Only a search per user.

We use the term **constellation** to refer to each one of the disjoint clusters in the network with common interests. Constellations may or may not be linked by **shortcut** nodes. Figure 13.3 shows an example of a clustered network. We focus our view on a randomly picked user that is defined as the center of the network, and nodes are organized in rings according to the number of network hops from the central user. Each of these rings is a neighborhood as in table 12.1. Nodes that are similar to the central user should be in the lower neighborhoods, and thus the search process is quicker and easier. The right side of figure 13.3 shows an example of a distribution of documents that are interesting for the central user. In this example, there is a “constellation” of documents at Γ^3. If the user limits the number hops from their messages to save network resources, this constellation of documents won’t be reached. There should be a mechanism to find the more-distant constellations without increasing the number of messages in the network.

As we will show later, as time goes by and the user learns about the structure of the network, the recall ratio goes up to nearly 90%. In any case, these figures suggest that the very first queries users perform in the network will obtain an extremely poor recall ratio compared to a simple flooding. Thus, flooding the network at least during the initial steps (that is, giving extra importance to the parameter M_{clus}) seems like a good idea. This approach is selfish, since it wastes network resources in the form of thousands of routed messages to get a single document. Figure 13.4 shows the number of messages that a single search sends to the network. If every user floods the network with several

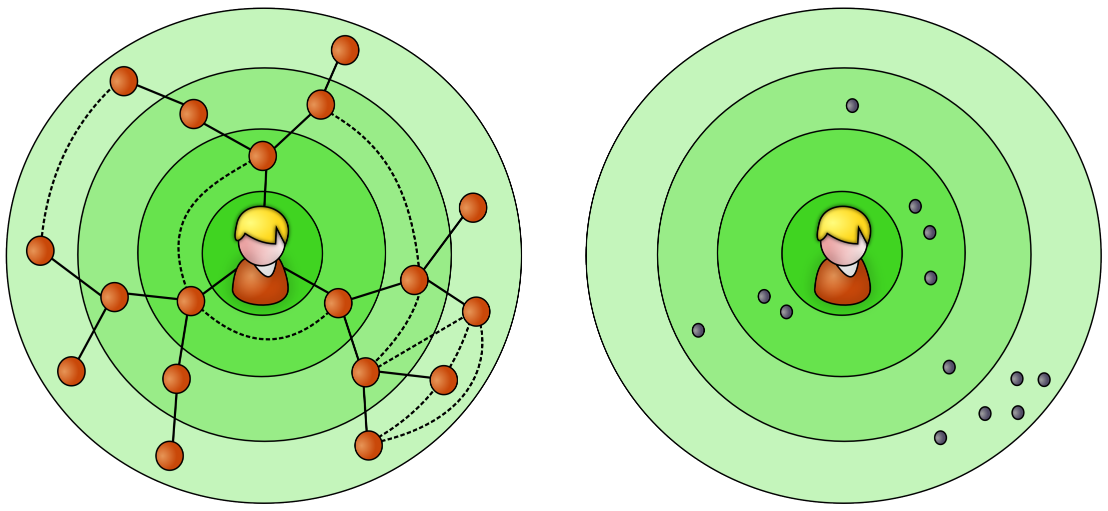


Figure 13.3.: The neighborhoods of a node and some constellations of resources

thousand messages to get a single recommendation, the network is likely to collapse very quickly. Furthermore, this behavior will prevent the creation of clusters and therefore it exhibits a slow convergence.

From this study, we conclude that the proposal has started to create a clustered network, even after a single search from every node. While nodes find documents only in their neighborhood Γ^3 at most (Figure 13.2), the current proposal exhibits a recall ratio lower than the ratio of flooding (Figure 13.1). We believe this is the result of the presence of unlinked constellations, communities of users that share interests but don't know each other (Figure 13.3).

Based on these data, we believe it is possible to enhance the results of the simple epidemic algorithms by means of helping to identify and locate faraway constellations. Mechanisms to help to this location will be presented in the next section.

13.3. Enhanced clustering of users

The algorithms detailed in section 13.1 may isolate two or more groups of nodes with the same interests, creating different constellations. In these cases, the epidemic search by itself may not be enough to connect two isolated groups. To prevent the creation of constellations, or at least to minimize their probability, we define a new structure based on classic DHTs. We will use the term **Soft Distributed Hash Table** (SoftDHT) to refer to this new structure.

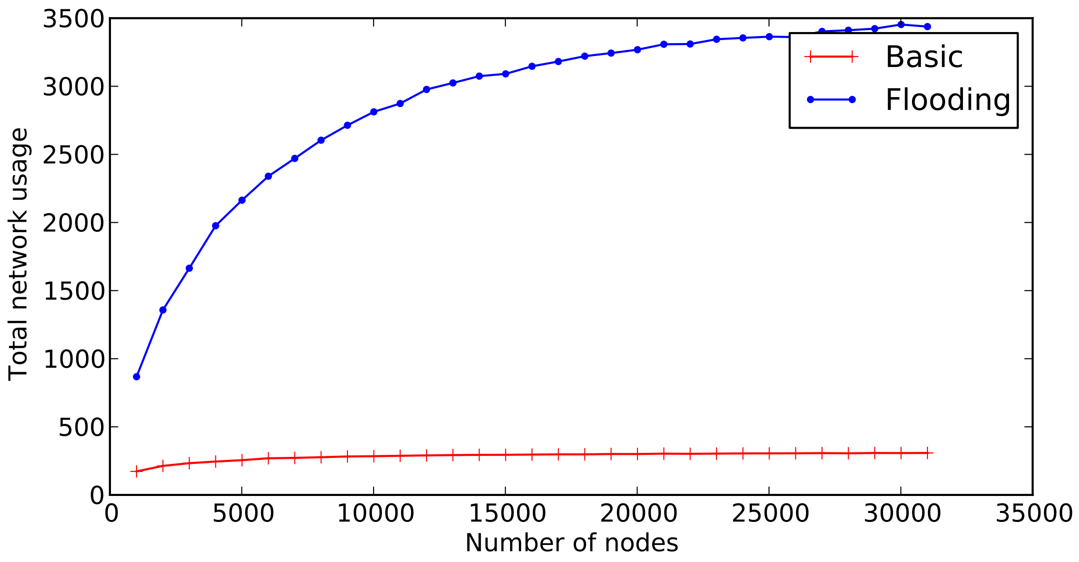


Figure 13.4.: Total number of hops (including w/o results)

13.3.1. Soft Distributed Hash Table

In a traditional Distributed Hash Table (DHT), a client uses a key to access a single piece of data. Any access with the same key will return the same data. DHTs are designed in such a way that the number of possible keys is much higher than the number of nodes in the DHT. Therefore, each node is in charge of a segment of the key space and data is spread through all available nodes. Some examples of DHTs are Chord [119] and Kademlia [88].

In our proposed SoftDHT, the number of keys is smaller than the number or nodes. In this case, several nodes are in charge of the same key and therefore different requests for a key may return different data. This kind of table is very suitable for our purposes and is simpler and faster than a standard DHT, since it won't have such tight constraints to keep data integrity as a normal DHT has.

In our implementation of the SoftDHT, we used a modification of Kademlia [88]. In Kademlia, each node stores k sets of addresses to n different nodes, called k-buckets. Each k-bucket stores only addresses that are under a certain distance from the original node identifier, using a xor metric. k-buckets work as FIFO queues, where addresses are removed when the corresponding node is no longer available. When a node searches for a key, it asks to the m nodes of the k-bucket that are closer to the petition. If any of them knows the response, they answer immediately. If not, they return the addresses of other m nodes that they believe that are closer than them to the identifier. This algorithm assumes that only those nodes that are closer to the key will store associated data and imposes integrity on these data. Kademlia is able to get an answer after contacting $O(\log(N))$ nodes.

We propose a modified DHT based on Kademlia. We configure the number of elements in each k-bucket (n) to be small enough to create fast rotations of addresses in every k-bucket even if the corresponding node is still on-line. Moreover, we configured a small m to force nodes to consult a small number of neighbors at once. Finally, we relaxed the restrictions of distance to the key to store a value and therefore data integrity is no longer warranted in a SoftDHT. That way, different inserts under the same key k may end in different subsets of nodes C_k. In this scenario, the value returned for a key k may be any of the values stored by the nodes in the associated subset C_k.

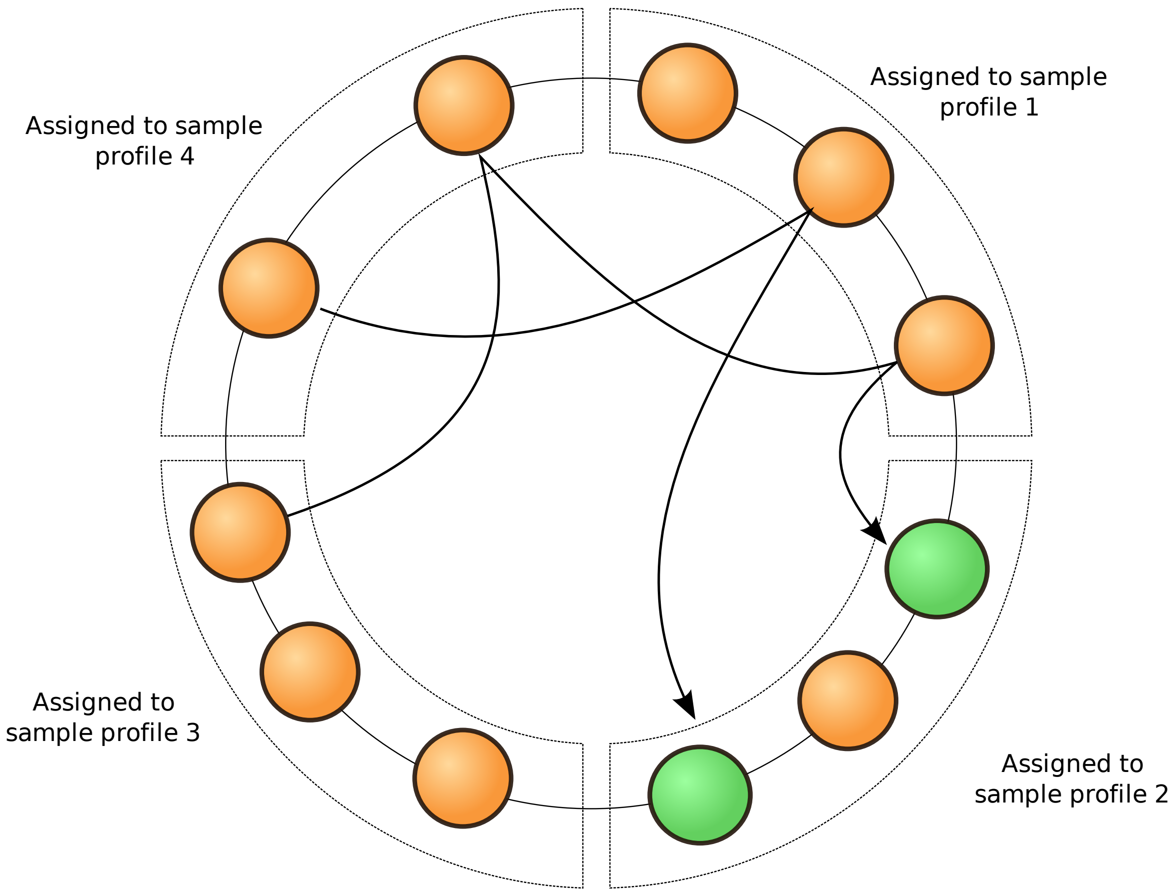

Assigned to sample profile 1
Assigned to sample profile 4
Assigned to sample profile 3
Assigned to sample profile 2

Figure 13.5.: Example of a SoftDHT

Figure 13.5 shows an example of a SoftDHT with 11 nodes and only 4 keys. Two different queries for the same key end in different nodes of the subset that is in charge of the key, and thus these queries will have different answers.

Next, we detail how we use the SoftDHT in our system. Let us define S as a set of well-known sample profiles in the network. The details about the construction of this set fall beyond the scope of this thesis. For example and for the sake of simplicity, S may be a set of profiles picked at random in the social space. All nodes in the network have access to the same set S. If we define the similarity of users as in section 12.2.1, any user in the network can compute their similarity to the sample profiles in S. In addition, a number of nodes –the complete network, or a selected subset– maintain a

SoftDHT, with S being set of keys of the SoftDHT. The cardinality of the set of samples, $|S|$, should be much less than the cardinality of the set of nodes in the SoftDHT, and therefore there are subsets of nodes that take part of the SoftDHT that manages the same sample description.

Every user in the network calculates and identifies which sample profile in S is their closest, and then they publish their address in the SoftDHT under the identifier of the closest sample description. Since several nodes in the SoftDHT manage the same key, the routing algorithm of the SoftDHT may finish in any of them. In other words, many nodes can publish their addresses under the same key and a request to the SoftDHT will return the address of any of them apparently at random. Since we are interested in saving as many nodes as possible under the same key, our proposed SoftDHT is able to have more aggressive caching methods than standard DHTs. Thus, paths are much shorter and nodes get results quicker than in traditional DHTs, at the cost of no data integrity at all.

13.3.2. Rules for new links

Besides the SoftDHT, some additional mechanisms are introduced in the basic algorithms to overcome the undesired behavior described in Section 13.2. We bounded the maximum number of links that a node have to other nodes (k). In a real system, nodes disappear from the network because they leave, cease to work or communication fails. Most peer-to-peer protocols have a PING mechanism to detect these disappearances, and thus (i) nodes will occasionally have empty slots in their neighborhoods that can be filled with new links, and (ii) a node will often find other affine users when it does not have any empty slots. Handling these two situations is the aim of this section.

Classical solutions to remove links such as first-in-first-out or less-recently-connected won't work very well in our proposal, since they do not take into account the importance of the link. In the social network that we are studying, a link between two users u and v has two important characteristics. The first of them is the similarity of the nodes in the link, $s(u, v)$. The second one is the relative clustering coefficient of the nodes, $\gamma_u(v)$. According to these parameters, some links are not often used but are still important to maintain a suitable social network. In this regard, we propose links to be managed according to the similarity and clustering of both nodes, not the actual use of the link.

If $\gamma_u(v)$ is high, it means that there are other nodes in Γ_u^1 that link to v. This way, removing the link $u \to v$ won't have a noticeable effect on the performance of searches, assuming that the maximum number of hops is $H > 2$, and therefore v should be available in Γ_u^2 through other node $w \in \Gamma_u^1$, $u \to w \to v$. In this regard, u can safely remove links to nodes with a high relative clustering coefficient. Similarly, a node w affine to u is probably close to other nodes in Γ_u^1 and it is likely at least in Γ_u^2, so u can safely remove w. Meanwhile, a node y with low relative clustering coefficient may be a shortcut to other constellations of the network, and since u is the only node in Λ_u^1 that links to y, removing y from Γ_u^1 may reduce the performance of searches.

We apply this idea as described next. When a node u receives a link request from a previously unknown node v, and u has its neighbor list complete, it calculates the affinity $s(\bar{u}, \bar{v})$ and clustering coefficient $\gamma_u(v)$ to the new node. Then, u checks these values against the median of the values in Γ^1_u. If $s(u, v)$ is less than the median affinities and $\gamma_u(v)$ is less than the median clustering coefficients, the new node is accepted. The less affine node in Γ^1_u is removed to make room for v.

Finally, a small-world network is a network where nodes of Γ^1 have a small parameter γ and yet the network has a small diameter. Small-world theory predicts that there is a low number of hops between any two nodes of the network, even in networks of thousands of nodes. As shown in [145], we can force a small-world network by means of creating random links in a highly clustered network. Thus, a certain number of random nodes are inserted both in the neighborhood of a node and in the list of next nodes to route messages. We will study this random criteria in the next section.

13.3.3. Enhanced algorithm definition

Next, we introduce the proposed enhancements to the algorithms described in the last section as pseudo-code.

Algorithm 9: join_network(λ), run by a new user

```
Data: New user of the network, $v$. Set of sample profiles, $S$
Input: $\lambda$, the similarity threshold
begin
    insert_random_links()
    foreach $w \in search\_user(v, \lambda, 1)$ do
        user_found($w$)
    $\bar{s} \longleftarrow$ most similar profile to $\bar{p}_v$ in $S$
    foreach $w \in SoftDHT.get(\bar{s})$ do
        user_found($w$)
```

The **user_found** of Algorithm 9 is significantly more complex than Algorithm 4. Now, we use the SoftDHT to let nodes discover other nodes that are close to their description. Periodically, a node searches for its nearest sample profile from S and will receive the address of another node that is close to the same sample description. This could be a neighbor that is already known, its own address or a completely new friend from a constellation located far away.

In addition, the simple neighborhood management of the original system is not enough. Algorithm 10 shows the procedure to decide whether or not a recently discovered user should be included in the neighborhood, when the maximum number of neighbors has been reached.

Algorithm 10: user_found(v)

```
Data: Current user $u$, and her attributes $\Gamma^1_u$
Input: New affine user found, $v$
begin
    if $v \in \Gamma_u$ then
        return
    if $|\Gamma^1_u| > k$ then
        $s \longleftarrow$ get_similarity($u$, $v$)
        $c \longleftarrow$ get_clustering($u$, $v$)
        $median \leftarrow |\Gamma^1_u|/2$
        if $d < order\_by\_similarity(\Gamma^1_u, \bar{p}_v)[median]$ and
        $c < order\_by\_clustering(\Gamma^1_u, \bar{p}_v)[median]$ then
            s $\leftarrow$ order_by_similarity($\Gamma^1$, $\bar{p}_v$)
            remove less similar element from $\Gamma^1_u$
            append $v$ to $\Gamma^1_u$
    else
        append $v$ to $\Gamma^1_u$
```

In the original version of the algorithms, if a new affine user was found when there was no empty slots in the neighborhood, the link to the less similar node was removed. Now, it has to handle two additional criteria: affinity and clustering. The joining is not unconditional, as we discussed in section 13.3.2, and the median affinity with the nodes currently in Γ_u must be exceeded. If the recently found node is accepted, the less similar node is unlinked.

In the next section, we will simulate all of these algorithms to test whether or not they enhance the search results.

13.4. Performance evaluation of enhanced clustering of users

This section includes thorough simulations of the system proposed in this part of the thesis. Recall ratio, number of messages sent, clustering coefficient and convergence are obtained by comparing the behavior of the protocol with a flooding algorithm.

Flooding was used in the past to find documents in P2P networks [71, 142]. Despite not being scalable with the number of nodes and the fact that searches consume considerable bandwidth, a message that is flooded arrives to a large portion of the network, ideally to all nodes. We consider flooding algorithms suitable for comparison with our proposal to analyze how well our system behaves.

In the simulations that follow, we define the **recall ratio** of the search algorithm as the average of the recall ratio for every search in the network. To calculate this ratio, we do not consider searches that returned no results, since this is a different parameter that we will study later. We have limited the maximum number of hops to $H = 5$ even for the flooding algorithm, and we use a social space of vectors of ten categories, $n = 10$. Additionally, the routing protocol should prevent cycles and our simulations include message identification to prevent cycles. If there is no limit on the number of hops, messages will travel continuously the network, eventually reaching every node and discovering all interesting documents. Simulations will show that, for networks up to several thousand nodes, the diameter of the network with randoms links is about 5 (see Table 13.1), and thus we consider this limit appropriate for our system. We model that users only share a single document with a random profile, and consequently, the user's profile and the profile of the document that they share are the same. Then, we create networks of different sizes where nodes join one by one, running the algorithm that was proposed in Section 13.1. To reduce the "scatteredness" of random user's profiles, the threshold λ for affinity definition is modified according to the number of nodes, $\lambda = \frac{2}{N^n}$, being N the number of nodes in the network. In our simulation, nodes perform searches one after another, and consequently the last node of the network has some advantages over the others due to the learning capabilities of the algorithms even before the first query. In between searches, nodes are shuffled to minimize this effect.

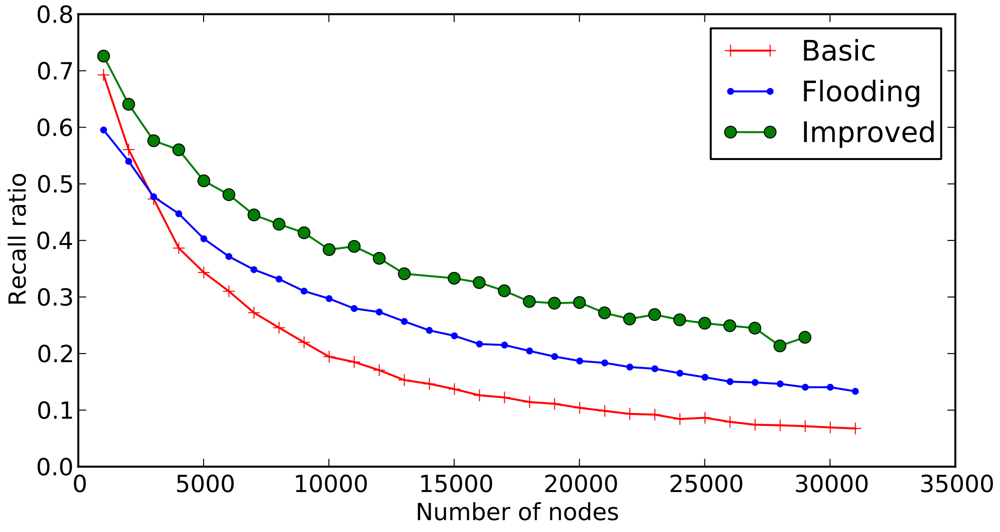
To run the simulations of this section, we have used a modification of Peersim [34] that is able to simulate networks with thousands of nodes.

13.4.1. First steps of nodes in the network

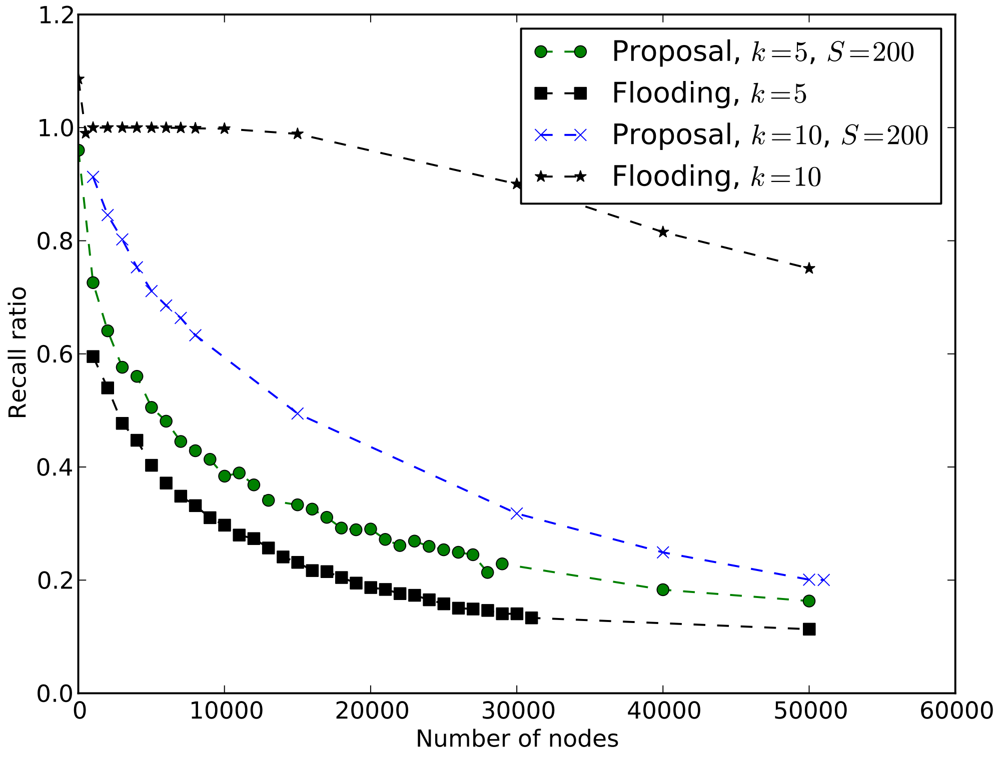
During this simulation we are interested in the recall ratio during the initial phases of the protocol, and so once the network is created every node performs a single search. During flooding, nodes forward the message to their k neighbors, while epidemics will choose only $M_{rand} + M_{sim} + M_{clus} \leq k$ neighbors to forward the message. During the initial steps, the additional messages handled by flooding create an advantage that may be not yet compensated by the improved knowledge of the network that the epidemic algorithm will have.

Figure 13.6(a) is the same as Figure 13.1, adding the enhancements that were proposed in Section 13.3.3 ($H = 5, k = 5, M_1 = M_2 = M_3 = 1$). The figure shows that the additional mechanisms not only improve the recall ratio of the searches over the basic algorithm, but in fact surpass the results of the limited flooding algorithm. In this case, we are simulating a SoftDHT of 10 sample profiles. This beating is possible thanks to the new structure that the SoftDHT created, since it is able to locate far constellations and keep documents of interest closer to the users. The right side of Figure 13.6(a) shows that in a network of $30,000$ nodes, the flooding algorithm limited to $H = 5$ hops can locate 20% of the documents that are of interest to a user, while our basic proposal is only able to identify 10% of these documents and after the enhancements, 30% of the documents. For the rest of the simulations of this document, only the results of the enhanced version of the protocol are included.

Figure 13.6(b) shows the dependence of the recall ratio of different network sizes and maximum number of neighbors k ($H = 5, k = 5, M_1 = M_2 = M_3 = 1$). Since there was only a single search, the lack of knowledge about the network made the recall ratio



(a) Recall ratio of basic, enhanced and flooding



(b) Recall ratio for different k

Figure 13.6.: Recall ratio for different network sizes and a single search

decrease when the network grew in size. Meanwhile, the high recall ratio when there is a low number of nodes in the network is caused by the flooding effect and the low diameter of random networks. Indeed, in a network with the specified parameters M_x, any node will contact up to 300 nodes for search. Hence, the likelihood of discovering interesting documents solely by means of flooding is high when the network size is under $1,000$. In any case, Figure 13.6(b) shows that the presence of the SoftDHT mechanism makes possible for the improved protocols to surpass the results of the flooding protocol, something that the basic version was not able to do as Figure 13.6(a) shows. Increasing the number of neighbors to $k = 10$, while maintaining the same value for parameter M_x, enhances the recall ratio of the protocol. In this case, flooding is unsurpassable (for $N < 10,000$, $k = 10$, the network diameter is less than $H = 5$ and all nodes of the network can be reached by flooding).

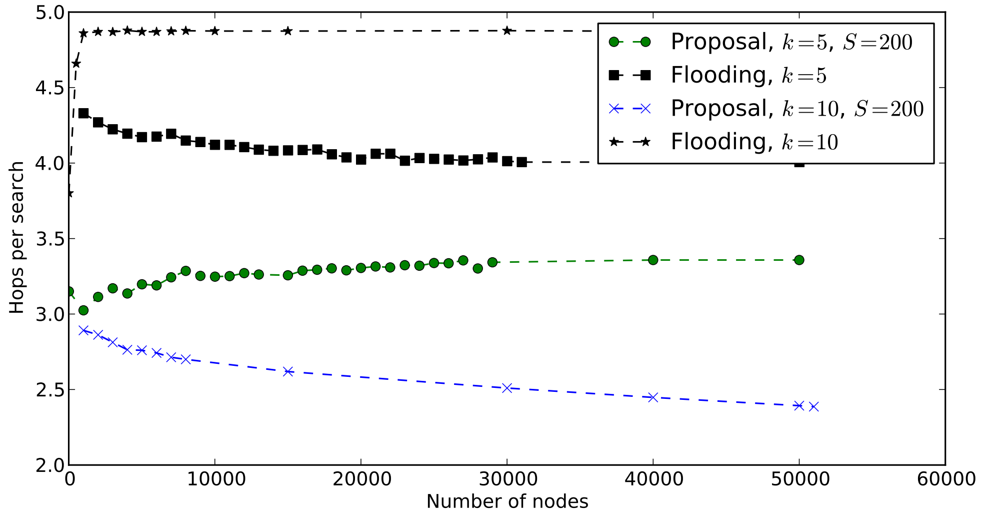
Next, we will analyze the use of the network for each algorithm. During the last simulation, we counted the total number of messages that the protocols sent through the network, and the average hops that a query needs to locate a document of interest. Figure 13.7 shows the results of this simulation.

It is interesting to compare the results in Figure 13.6(b) with Figure 13.7(a). The average number of hops for the epidemic algorithm is close to 3, for the scenarios of $k = 5$ and $k = 10$. This means users only find documents in their own cluster of interest defined as Γ^3 and similar users are close in networks hops even after a single search. One advantage is that the low number of hops means that the discovery is extremely fast in this algorithm, and that it is scalable with the number of nodes in the network.

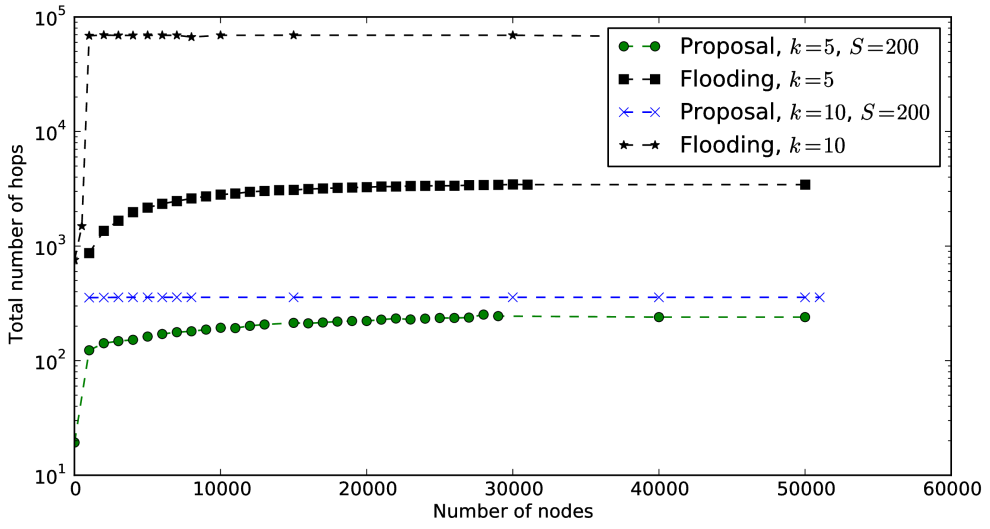
Figure 13.7(b) shows the network use of both protocols in a logarithmic scale, defined as the number of hops needed to perform a search even when there is no result. This figure shows that network use slightly increases with the number of nodes until a limit is reached, and thus that the epidemic algorithm is scalable in N. The original epidemic algorithm sends more messages to the network than the enhanced version, since it knows fewer similar nodes. These extra messages used by the basic algorithm are the result of a low-clustered network, but the additional use of the network is not enough to achieve a better recall ratio. The proposed modifications are still scalable with the number of nodes, and even improve upon the network use of the basic proposal.

Figure 13.8 shows, for a single search, the percentage of searches with no results for an increasing number of nodes in the network. In order to test whether or not the algorithms can find rare items, the number of documents in the network that match a rule is intentionally low during this simulation. The high number of searches without results for the basic algorithm is notable, while the proposed modifications enhance the recall ratio, since they are able to find remote constellations. In any case, the results for the clustered algorithms can be enhanced by an increased number of searches, as we will see next. Figures 13.6(b) together with Figure 13.8 show that the quick convergence of the modified algorithm makes the proposal highly recommendable in a scenario of few searches or when many new users join the network.

Finally, Figure 13.9 shows the recall of the algorithm with an increasing number of searches in a network of $10,000$ nodes. When the number of searches increases, the knowledge of the network of each node increases as well, and consequently the recall



(a) Number of hops per search



(b) Total number of messages

Figure 13.7.: Network use of the search protocol

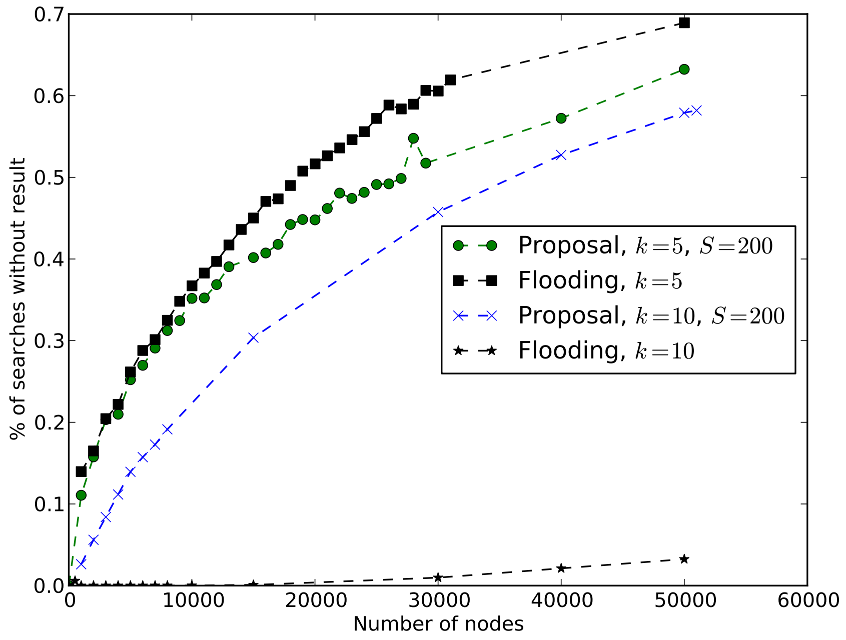


Figure 13.8.: Searches without results

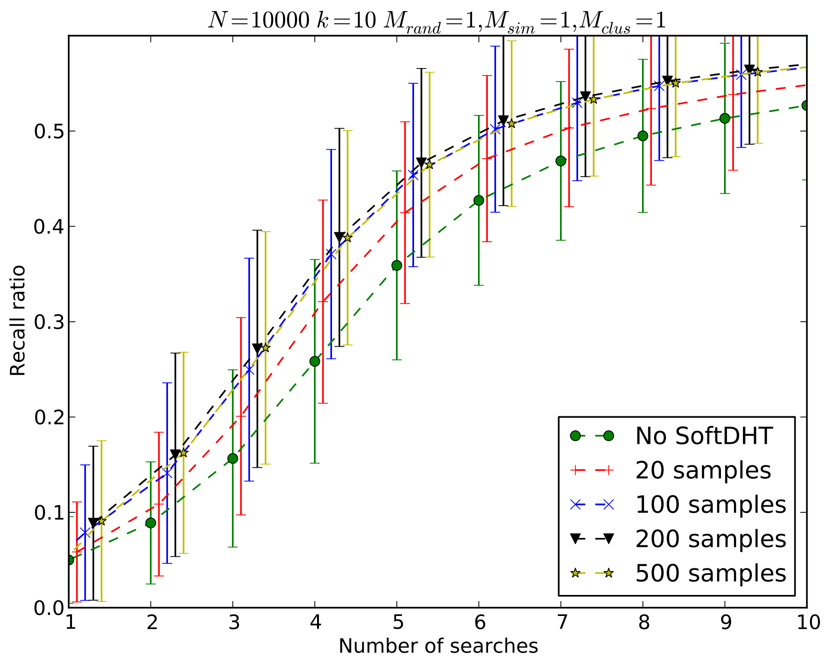
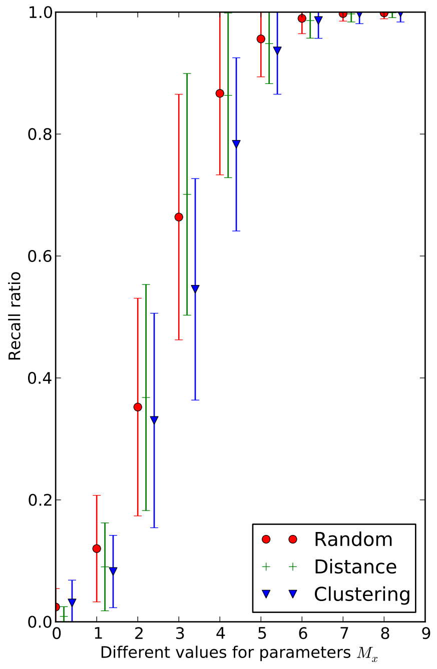
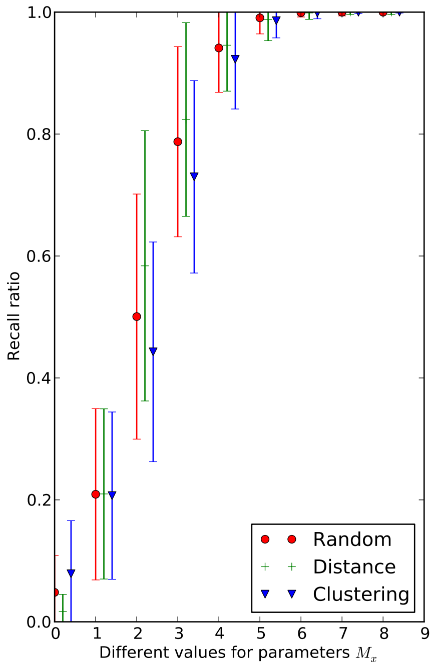


Figure 13.9.: Evolution of searches



(a) $N = 10,000$



(b) $N = 5,000$

Figure 13.10.: Recall ratio for several values of M_s

ratio of searches is higher. The recall ratio is over 60% of matching documents in the network after only ten searches in this scenario.

13.4.2. Setting the different parameters of the protocols (M_{rand}, M_{sim}, M_{clus})

Figure 13.10 shows how the recall ratio of protocols varies according to the different parameters M_{rand} (random), M_{sim} (similarity) and M_{clus} (clustering) for two networks of $N = 10,000$ and $N = 5,000$ nodes, and $k = 10$. In this simulation, two of the parameters are fixed to $M = 1$ and the one being studied varies from $M_x = 0$ to $M_x = 8$. Then, we wait for the stability region before calculating the recall ratio. The figure shows that the recall ratio of the algorithm does not depend on the exact combination of parameters M_{rand}, M_{sim} and M_{clus}. Since these parameters determine how many neighbors are contacted in each step of the route, a higher parameter means more messages in the network and thus the recall ration of the protocol gets closer to the recall ration of flooding.

Figure 13.10 does not capture how recommendations are retrieved. At first glance, the recall ratio achieved by the protocol does not depend on the value of M_x: regardless of how we choose the next node in the path, we will get a very similar recall ratio. Despite

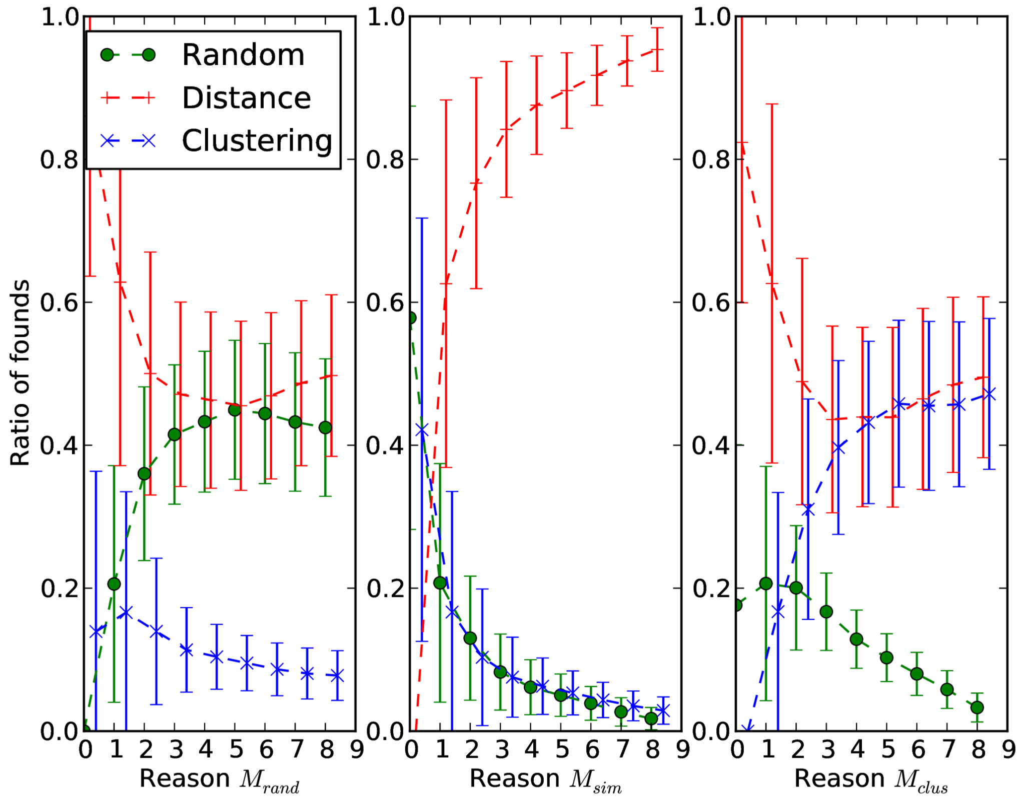


Figure 13.11.: Percentage of results according to the several routing rules.

this fact, it is important to learn how the recall ratio is achieved. Figure 13.11 shows the percentage of the recommendations received by a user as a result of the messages that they sent according to the several routing rules. For example, 50% of the recommended documents were found using messages routed through the similarity criterion, even when $M_{rand} = 5, M_{sim} = 1, M_{clus} = 1$. Figures 13.10 and 13.11 should be examined together. A close examination of these figures concludes that the random routing rule improves the recall ratio because it lets a fast location of affine neighbors to use the similarity rule with them later. Especially significant for this conclusion is the left side of Figure 13.10-a: increasing the number of messages sent to random neighbors decreases the documents that are found using this rule, but since the overall recall ratio increases, we can conclude that the network is better clustered now.

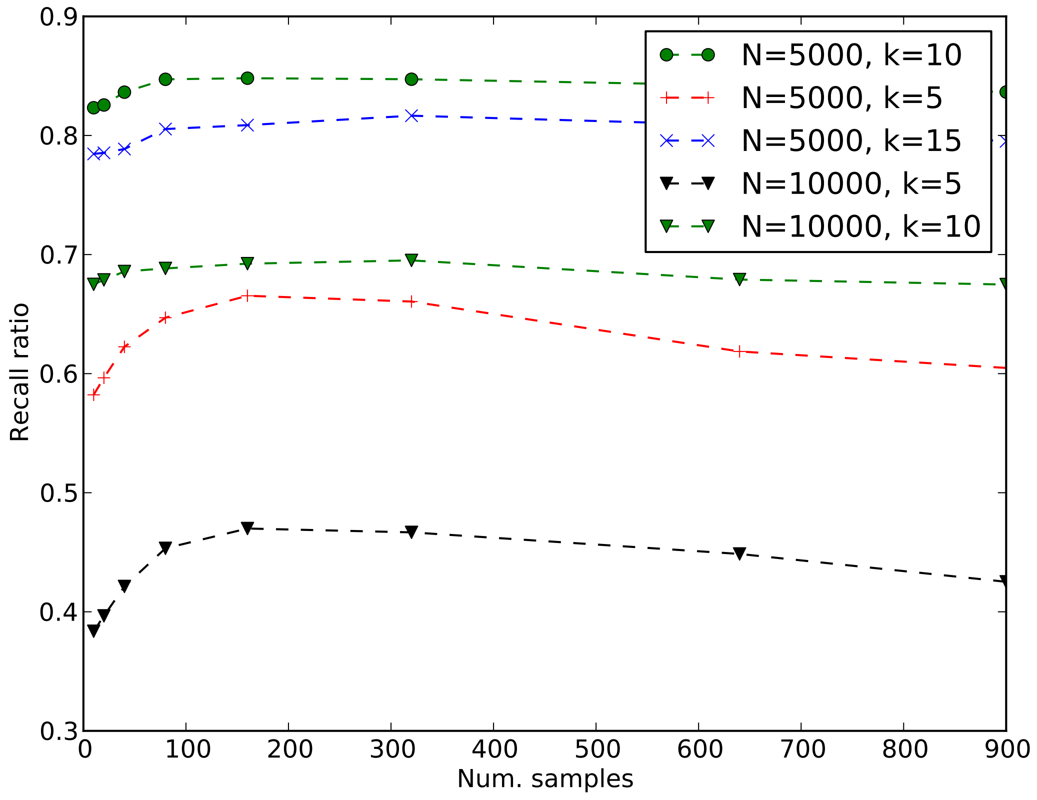


Figure 13.12.: Recall ratio to several sizes of the sample set

As we showed in previous simulations, the use of a SoftDHT enhances the recall ratio of the protocol during the initial phases and enables a faster convergence to the stable phase. Figure 13.12 shows the performance of the algorithm while varying the number of samples in the SoftDHT for several networks. As the figure shows, a low number of samples is enough to greatly improve the results of the network. As the number of samples increases, finding new constellations of nodes is increasingly harder: there

Table 13.1.: Network structure. $N = 5000$ $k = 10$ $M_{rand} = M_{sim} = M_{clus} = 1$ $Samples = 200$

	Cluster				Random			
i	$\|\Gamma_v^i\|$	$\|\Lambda_v^i\|$	$\%nodes$	$\%res$	$\|\Gamma_v^i\|$	$\|\Lambda_v^i\|$	$\%nodes$	$\%res$
1	10	11	0.22%	52.63%	10	11	0.22%	0.00%
2	14	25	0.50%	100.00%	89	100	2.00%	0.00%
3	54	79	1.58%	100.00%	746	846	16.92%	15.79%
4	169	248	4.96%	100.00%	3276	4122	82.44%	89.47%
5	455	703	14.06%	100.00%	878	5000	100.00%	100.00%
6	1199	1902	38.04%	100.00%	-	-	-	-
7	1759	3661	73.22%	100.00%	-	-	-	-
8	1205	4866	97.32%	100.00%	-	-	-	-
9	131	4997	99.94%	100.00%	-	-	-	-
10	3	5000	100.00%	100.00%	-	-	-	-

are more sample profiles, and fewer real users can identify with them. The use of the SoftDHT is significantly more important if the number of neighbors is low: even if the SoftDHT always improves the recall ratio of the algorithm, this improvement reaches 10% when $k = 5$, and 3% when $k = 10$.

13.4.3. Analysis of the clusters

In this section, we analyze how well clusters are built inside the social network using the mechanisms of this research.

Table 13.1 shows the spread of documents using the enhanced algorithms and compares it to a network with random links. This table shows the suitability of the construction algorithm for keeping potentially interesting documents close to the nodes. For this simulation, there are $N = 5,000$ nodes in the network, each one connecting to $k = 10$ other nodes. $M_x = 1$ and a sample space of $S = 200$. Then, we wait until every node in the network performs at least 4 searches. At the end of the simulation, we choose a random node u and calculate the cardinality of its neighborhood and distribution sequence sets, and the recall ratio for each hop, that is, for each Γ_u^i. We repeat this process with a network of the same size where nodes are randomly linked.

As Table 13.1 shows, 25 nodes are in Λ^2 and they hold 100% of the interesting resources. Our central node, in the worst case, only has to contact 25 nodes to get all of the interesting resources from the network. It has direct contact with 10 of them, which share more than 50% of the interesting resources. In contrast, a network with random links needs to contact the 846 nodes of Λ^3 in order to start receiving recommendations. Even in this case, it is only able to get about 16% of the interesting documents.

These numbers come at a cost. The highly clustered network in our proposal has a network diameter of 10 hops to visit the whole network. The random network is able to visit the $5,000$ thousand nodes in only 5 hops. The clustered structure of the social network is thus inefficient if the interests of the users change quickly, or if they frequently ask for recommendations for documents that are far away from their profile.

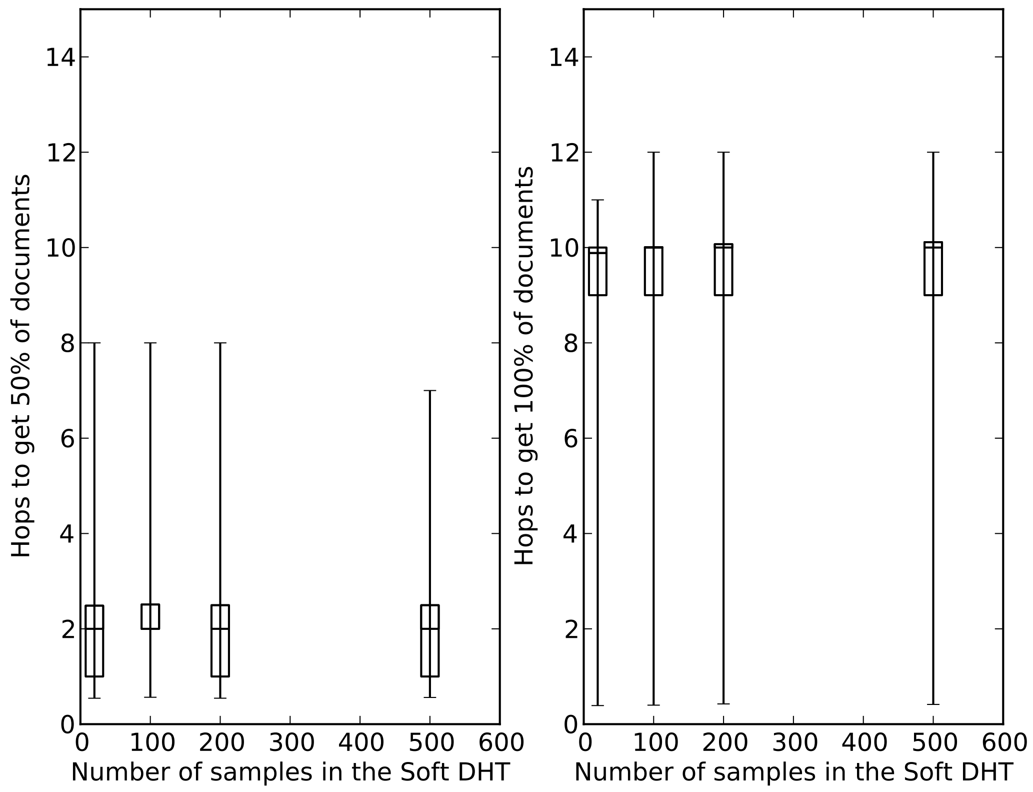


Figure 13.13.: Structure of the network

Table 13.1 shows the results for a single, randomly chosen node. Figure 13.13 extends the analysis to the whole network. In this figure, a network of $N = 10\ 000$ nodes is analyzed for $k = 10$, $M_{rand} = 2, M_{sim} = M_{clus} = 1$. The figure shows the upper, lower and second and fourth quartiles of the number of hops where 50% and 100% of the interesting documents were found. It shows that, with a slight dependence on the sample space, 50% of the interesting documents are in Γ^2. To get 100% of the interesting documents, users need an average of 10 hops.

Finally, Figure 13.14 shows the clustering coefficient for several network sizes and a varying size of the SoftDHT sample set. It shows that, for a network size smaller than $20,000$ nodes, having 200 sample profiles in the SoftDHT optimizes the clustering coefficient. Optimizing the clustering coefficient means that more affine nodes are connected, and therefore interesting documents are at less hops. Simulations of Section 13.4.2 show this. Figure 13.14-b shows the same data as 13.14-a, but with different axis. The right side of this figure shows that increasing the sample space is not always advisable: the clustering coefficient for a $N = 1,000$ network drops if the sample space is over $S = 200$.

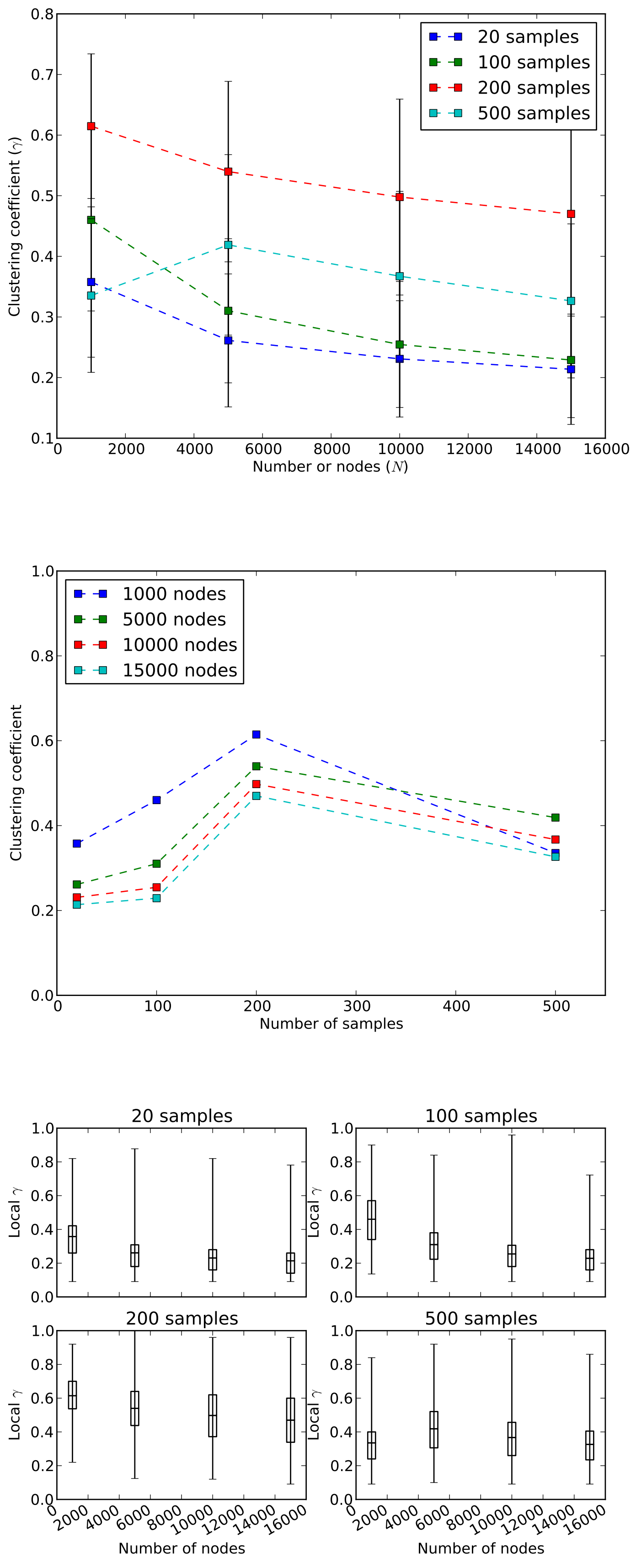


Figure 13.14.: The clustering coefficient

14. Conclusions

In part III, we justified that users are organized as a social network to aid in the process of protecting the participants of the system. In a social network, nodes link each other according to their affinities. We believe that the social network, at first a requirement of the security mechanisms, can be used to improve the results of the recommendation process. Indeed, the fact the nodes in a social network link to other nodes that are controlled by similar users enhances the results of the recommendation process and aids in the location of new and interesting documents.

In this part, we analyzed how epidemic algorithms can be used to locate interesting documents in a social network where links depend on the user's preferences. To achieve this, we define users, queries and documents as vectors in the same social space, and describe a common metric to calculate the similarity of documents as well as users.

The substrate of the network structure is created based on the affinity between the users' profiles, with enough random links to induce a small-world behavior. From this point forward, we can create new links and discover new user communities by taking advantage of the search results and the small-world behavior of social networks.

One desirable characteristic of the mechanisms to create a social network is that they must be fast and efficient. That is to say, the social network must help users to locate other users that are similar to them, not only when the user joins for the first time, but in every moment of the process.

We showed that the criteria of similarity, clustering and randomness do not improve enough the results of the epidemic algorithm. Hence, we proposed a SoftDHT, a structure of sample user profiles that aids in the location of islands of similar users that were not identified at first. In addition, we discussed that there is little gain if a node links to another node that already belongs to a highly clustered affinity group. It may be more useful to limit the number of neighbors and include links to other less clustered, external nodes. In this case, external and unknown groups of users that share the same interests but have not being discovered yet can be found.

These ideas and improvements were tested in section 13.4. We performed throughout simulations, and we found that the proposed enhancements improve the performance of basic epidemics algorithms in dynamic scenarios and shortens the convergence time, while having a comparable performance in the long run. In addition, we tested the network structure of the basic searching algorithm and the improved version, and found that the improvements aid in the creation of a network that shows the desired small world behavior.

Part V.

Confidentiality, integrity, persistence and availability

Part V: Confidentiality, integrity, persistence and availability

As we discussed in part I, getting a recommendation from the system is not enough. Recommender systems are usually involved in many of the most successful e-commerce shop of the internet, such as Amazon or IMDB. The main objective of an online shop is that clients buy and access the recommended products. In this regard, we believe that providing a mechanism to access, download and/or stream the recommended document is necessarily the final step of any recommender system. In addition, the final output of the process may be useful to enhance future recommendations, for example, if the user's profiles are based on their buying habits.

Accessing documents show some problems from the point of view of security that a secure recommender system must face. Indeed, any protection that we set on the profiles that a user gets as recommendation from the system is useless if the final access to the document can be traced down to the user. In addition, nodes of the network that store and provide access to the final documents are under threat of legal attacks, as we saw in part I.

In this part of the thesis, we introduce our contributions to the protection of users during the insertion and download of documents into and from the system.

15. Secure Cooperative File-system

In the information economy, data are one of the most valuable possessions for people and enterprises. In fact, it is of such importance that there is often a significant economic and/or sentimental damage if data loss occurs. This undesirable situation can be caused by human or computer errors, fire or water accidents or even both business and common burglars that steal hardware. The use of current distributed schemes such as P2P networks and cloud computing helps to minimize the risk of data losing. In addition to this helpful “backup service”, distributed environments let access the same documents from any of the devices that their owner uses, or from devices that are controlled by users that are different from the original owner of the documents.

During this thesis, we are developing our recommender system on cloud computing. As introduced in part I, we believe that accessing documents must be part of the recommendation process, and thus it is the last step that a user follows in the system. The distribution of files in a truly P2P network has many advantages that are suitable for our scenario.

- The distribution of documents among many nodes in the network protects data against local malfunctions or misbehavior that tries to prevent the access to the document. This way, the distribution of documents enhances the **document availability**.
- Distribution of documents in external nodes let that the document is available even if the publisher of the document is no longer online. Most distributed file-systems, hence, provide **document persistence**.
- Distribution of documents also means distribution of responsibility. Users that publish their document in a distributed network that cannot trace the source of a document are safe against the kind of legal attacks introduced in part I.
- It is easier and much cheaper the creation of distributed networks than the creation of a centralized node to distribute large sets of documents.

In addition, distribution of documents in a P2P network has some drawbacks in regard of the data security and user’s privacy.

- Many people will object if their data is stored in the computer of a complete stranger Even collecting the name of the files that a user has in his personal namespace is a violation of his privacy.

- The pervasive nature of distributed documents make difficult delete them when the user decides that they are no longer necessary.
- The open nature of many of these distributed networks makes difficult limit the access and download of documents only to selected users or group of users.
- Finally, regarding the deniability problem that we explored in part III, nodes that store dangerous document could be prosecuted if they are aware of the documents that they manage and provide.

In this chapter, we center our efforts in a distribution of documents in a P2P file-system that takes into account these challenges.

15.1. Related Work

There are many proposals on distributed file systems in the literature. From NFS [63] to AFS [64], many widely deployed distributed file systems rely on several replicated servers. This approach is no longer scalable nor cheap, and centralized servers are a honey-pot for attackers, even if they are replicated. To solve these problems, distributed filesystem over peer-to-peer networks appeared in the literature. In this section, we will summarize some of the properties of current and past distributed and decentralized filesystems that show a strong focus on security. Readers can consult a general analysis of distributed filesystems in [57].

MojoNation [1] was a robust, decentralized file storage system. Nodes were organized in a peer-to-peer network that showed a ring shape. Files were broken down in “pieces” or “blocks” that were replicated and stored in the network using a unique identifier linked to the file. MojoNation was a commercial filesystem, and the company that run the network made profit commercializing the cash units used in the distributed network, called *mojos*. Mojos were used to prevent abuse and document flooding, and users of the system tread their mojos to replicate the pieces of their files in other nodes of the network. MojoNation died because original designers didn’t consider the high rate of joins and leaves of common nodes.

Free Haven [37] uses a non-structured peer-to-peer network to organize nodes and routes messages that search for documents by flooding the network. In Free Haven, files are broken down in blocks using an information dispersal algorithm [101]. After joining the network, users create a pair of asymmetric keys for each file. Each block is indexed with the same key, $hash(PK_{sub})$, and traded with a previously computed and static list of neighbors that takes into account reputation, in the form of analysis of the past behavior of these neighbors. When clients want to download a file, they ask the network for its associated key, wait for enough blocks and reconstruct the original file. Nodes trade blocks with other nodes to improve anonymity and persistence of data.

FreeNet [27] was designed to store documents and allow its later access by means of an associated key. The goals of FreeNet include the prevention of censorship and offer anonymity both to the user who publishes the document and to the user that access it.

To achieve these goals, the Freenet network creates a no-hierarchical and no-structured organization of nodes that anonymously transmit and cache messages and documents among them.

GnuNet [123] has the objective of building a broadcast routing algorithm based on specific valuation of neighbor nodes. Each node valuates its neighbors based on their behavior and the number of messages that route or ask to route. Messages from less valued nodes receive less priority in the output queue. Each node routes its messages based on its own interests, but the economics of the network supports strong collaboration.

Over this routing algorithm a file sharing protocol was developed [54]. Files are divided in different blocks that are individually encrypted using their hash as the symmetric key. Then, the hash of the key is used as the file identifier. Nodes trade these blocks with neighbors, and get other blocks in exchange. A searching service is provided by means of introducing keywords. When users want to search for a file, they send the hash of the keyword to get the data blocks.

Shark [15] splits files in blocks and uses a distributed hash table (DHT) to store blocks and iNodes. When a node downloads a block, it publishes itself in the DHT as holder of a replica. Only nodes that are controlled by the same user will be proxies of personal data, so clients need connect to at least one of their personal nodes to retrieve their data. Clients must prove knowledge of the contents of a chunk to read/write, and then Shark is not able to provide anonymity.

Cooperative File System (CFS [32]) is a file system over a distributed hash table. Since this filesystem is going to be the base of our proposal, it will be explained in detail in the next section.

15.2. Distributed documents for a recommender system

We will use the filesystem developed in this chapter as the last step of the recommendation that a user gets from the system. Indeed, the recommender system does not output a profile or a link to the desired document but the real document. In this section, we will simplify this scenario to minimize the dependency with other parts of the recommender system. In any case, the reader should be aware that this simplified scenario does not modify the aim of this thesis, i.e. the development of a secure recommender system.

The simplified scenario of our study is as follows. Individuals want to use the internet as a backup of their personal documents, or as a nearly unlimited disk space for their data. In addition, they wish to provide access to these documents from any of their devices, or to other users that have the necessary links to the documents. In addition, they are really interested in keeping their documents private and out of the reach of both casual and commercial eyes.

The features of structured Peer-to-peer networks are very convenient to achieve reliability and accessibility of personal data to thousands of users at the same time. Among the networks that were studied in section 15.1, CFS is the only structured Peer-to-peer network. Authors of CFS didn't consider security in their original design, and this section describes the construction of a secure structure over a CFS-like system. We call this system Secure Cooperative File System (SCFS).

15.2.1. Cooperative File System

Our proposal aims to add security to the Cooperative File System (CFS [32]), which is a file system over a Distributed Hash Table (DHT). Figure 15.1 shows the architecture of this system.

A DHT is a structured peer-to-peer network where nodes pick up a random identifier ID_i. Each one finds and links at least to the node that has the very next $ID_n > ID_i$ in increasing order. The node ID_i is in charge of storing data that is identified with an $ID \in (ID_i, ID_n)$. The process goes on until that the last node links to the first one and the whole network creates a ring structure. In order to put or get the information identified with ID, a node sends clockwise a message in the ring to the node in charge of ID, which answers. To improve the routing in the ring, nodes have far links to other nodes of their choice. Current implementations of a DHTs include Kademlia [88] or Chord [119], and the main difference between them is the specific algorithm for far links.

CFS was implemented over Chord. It gets a file F with a file name f, divides the file in blocks, stores the blocks B_i and calculates H_i and ID_f.

$$F = \{B_1 \cup B_2 \cup B_3 ... \cup B_n\} \tag{15.1}$$
$$H_i = hash(B_i) \tag{15.2}$$
$$F_h = \{H_1, H_2, H_3, ..., H_i\} \tag{15.3}$$
$$ID_f = hash(f) \tag{15.4}$$

Then, CFS saves each block B_i in the DHT identified as H_i and stores the list of H_i in a special block under ID_f. Files in CFS are persistent and the system warrants that every file will be retrieved, regardless of its popularity.

15.2.2. Security Requirements

Distribution of personal data in peer-to-peer networks has many drawbacks from the point of view of security. Many people do not want that strangers were able to access to their private data, and even they will object if it is possible to get the knowledge of the existence of some files. Intruders may look for common file names as "accounting", "strategic plan" or "passwords" in the whole network for any legitimate user. In addition, in some countries having a single MP3 file or some kind of adult content may be severally prosecuted.

The security requirements of SCFS are:

Confidentiality Data should not be readable by others apart from the user that uploaded it. In a distributed filesystem every node stores personal data from any user, and the filesystem must supply mechanisms to warrant that node administrators cannot read data from other users. In addition, if nodes cannot read the content that they store

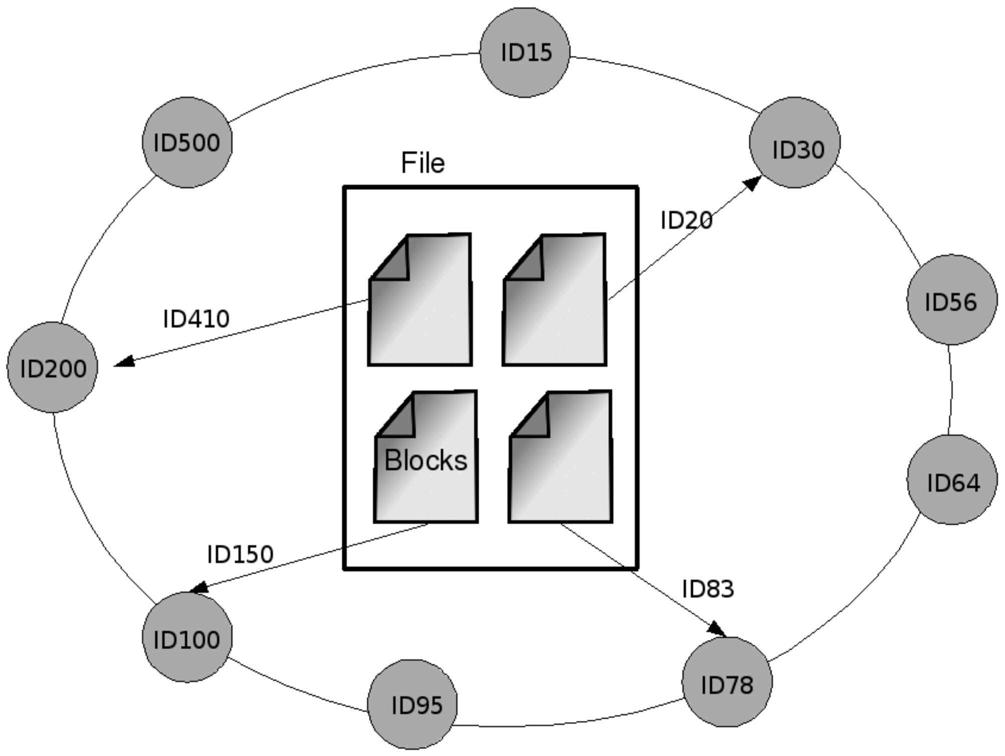


Figure 15.1.: Cooperative File System.

they can positively deny its knowledge and protect themselves from legal prosecution for storing and distributing illicit content.

Privacy Malicious users may collect data about habits or interests of users. Even if an attacker is not able to access to actual data, the name of the files of the user name space may be relevant. Commercial research and dictatorial governments may get profit from capturing the names of files accessed by users. The filesystem must provide methods to prevent such eavesdropping.

Integrity Since data is stored in uncontrolled nodes, it is not possible to prevent the modification of data. The filesystem must provide mechanisms to detect modification and restore original data, if possible.

Persistence The filesystem must prevent data loss. Files can be lost by means of malicious nodes that remove pieces of data, users that write data in the same place, both intentionally and unintentionally, or network or node failures. Storage systems focus on persistence instead of publishing the data.

Availability Users are in the move and own many devices with different network capabilities, memory and processing power. In spite of this, they want to access to a consistent disk space from every device regardless of its capabilities. The system should provide mechanisms to allow data to be available from any of the devices of the users, regardless how and where they are connected.

In the next section we propose several mechanisms on top of CFS to cover these design objectives.

15.3. Secure Cooperative File System

Distributed file systems over structured peer-to-peer networks provide the requirements that the scenario under study needs. CFS is the first choice to develop a distributed file system for personal files, but it does not provide the necessary security services. In this section, we propose some mechanisms to add to the standard CFS system in order to cover the security objectives studied in section 15.2.2.

This section will explain in depth each of the steps. A graphical outline of the whole process is shown in figure 15.2.

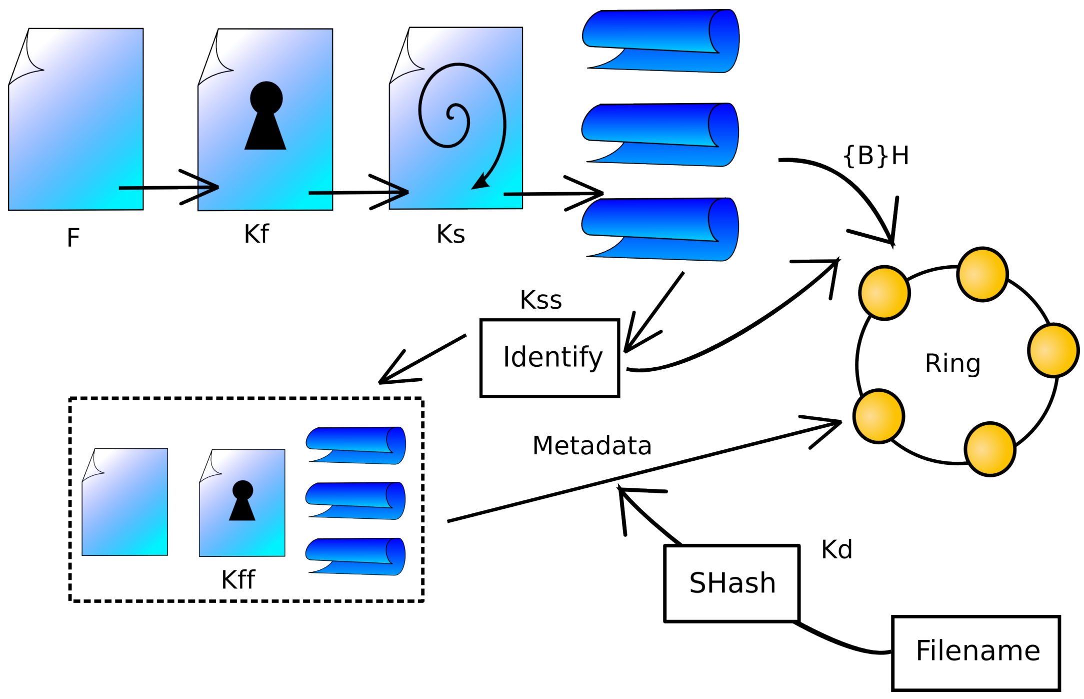


Figure 15.2.: Description of the process.

15.3.1. Assumptions and Definitions

Table 15.1 includes the definition of the main concepts of SCFS. Readers will find a relation of symbols in table 15.2.

In order to organize all this information in a single point, configuration file exists. A configuration file may be associated to a file, a directory and its contents or every single document of the user. Configuration files store the identifier of the root directory of the user, if needed, and the set of keys K required to get the associated object or set of objects. These configuration files are stored locally to the user in each one of the nodes and kept private.

Table 15.1.: Elements of the distributed filesystem

User	A human client of the system.
Node	Each one of the devices that a client uses to join to the network.
File	An ordered array of data
Directory	An unordered set of files and directories
Filename	A human readable identifier for a file or directory. It is not unique in the system, since two different users can store different files under the same filename
Root Directory	The filename of the directory that holds every file and directory of the user. Although not mandatory, we assume that users will have a root directory for all his documents and files.
Block	Each one of the pieces than the IDA creates from a file
Metadata Blocks	Special blocks of data that have enough information to restore the complete file
iNode	The first block of metadata that holds a list of the rest of metadata blocks

For example, Bob and Alice join to a SCFS network. They both desire to store a file that has "revenues.xls" as filename. Bob has $UID_{bob} = bob$ as his identifier, and Alice $UID_{alice} = alice$. Bob has $cfs : //bob/root/$ as the root directory, and Alice has $cfs : //alice/root/$. In order to access to the personal root directory of Bob, he will need his global key K_d. Since their user identifiers and K_d are different, their name-spaces will be different as well, as the next section shows.

15.3.2. Securing the Whole File

The first step to secure a file is encrypting its contents. Each file is encrypted with a symmetric algorithm with K_f. Encrypting a file ensures that casual attackers won't be able to sniff its contents.

After the encryption process, a shuffling and redundancy creator algorithm takes place. SCFS uses an information dispersion algorithm (IDA) described in [101]. This step has a double objective. The first one is preventing that consecutive bytes in the original file were consecutive in the stored blocks. This shuffling prevents some kind of statistical attacks against files with a known structure or common header, such as PDF files. Since users can choose the kernel space of vectors for the IDA algorithm, the spreading of data in the final blocks is deterministic only for the original author. Key K_s is used to generate the vector space that the IDA algorithms need. The second objective of the algorithm is creating redundancy of data. In order to enhance the availability of the service, the algorithm takes the original file and create m blocks in such a way that it will need $k < m$ blocks to restore the original file. During the reading process that redundant information may be used to check the integrity of data and correct errors. In this regard, the system is protected against malicious nodes that randomly change the data that they store, overloaded nodes than cannot manage all their parallel connections and local network failures. The dimension of B_i is chosen in such a way that it matches

Table 15.2.: Symbols used in SCFS

UID	The identifier of each user. UID is the same for every node that a user controls.
F	The original file of the user
ID_f	The final identifier of a file
S	The encrypted file of the user
B_i	Each one of the blocks of a file
B	The set of blocks of the file
H	The set of identifiers of blocks
K_d	The key used to secure the filename
K_f	The key used to encrypt the file
K_{ff}	The key used to encrypt the metadata
K_s	The key used to generate a vector space for the IDA algorithm
K_{ss}	The key used to create block identifiers
K	The set of keys. It can be generated from a master key associated to a single file.
N	Number of nodes in the network
M	Number of malicious nodes in the network
B	Number of blocks per file
b	Number of necessary blocks per file
k	Average hops in the P2P substrate. $k = \log(N)$ in Kademlia

the size of a block in the DHT.

$$S = \{F\}_{K_f} \tag{15.5}$$
$$B = \{S\}_{IDA} = \{B_1, B_2, \ldots, B_i\} \tag{15.6}$$
$$dim(F) = dim(S) < dim(\cup B) \tag{15.7}$$

Clearly, this algorithm consumes time, bandwidth and disk space in remote nodes since the dimension of the final data in B is greater than the dimension of the original file F. Users can choose the amount of redundancy to apply to each file, or even if this algorithm runs or not at all over their files. Authors expect that clients will use most of the time the default redundancy of 30%.

The encryption and shuffling steps take place locally in a node and the set B is not yet published in the ring.

15.3.3. Securing File Blocks

After the previous step each block B_i has the same size than the DHT blocks. Since the data in each B_i was encrypted and shuffled, it makes little sense to any casual attacker that sniffs the blocks.

The CFS stores and retrieves blocks of data associated to an identifier. We propose three different methods to identify each one of the blocks. All of them have their advantages and disadvantages.

Random identifiers. The local node picks up a random identifier for each block. The identifier has the same size than identifiers in DHT. This is the most secure method since the random identifier has not information about the block, its contents or its publisher. In addition, the set of ordered random identifiers of the file blocks must be stored by the user locally.

$$H = \{random_i\} \mid dim(H) = dim(B) \tag{15.8}$$

A file of 20MB that is stored in blocks of 2048B need about 160KB to store the random identifiers. Since SCFS uses special blocks to store file identifiers, this approach needs at least 80 blocks just to store the file structure.

Pseudo-random identifiers. Identifiers for each block are created with a pseudo-random algorithm using $K_s s$ as seed of the algorithm.

$$H = \{K_{ss}\} \tag{15.9}$$

In this case, the user only has to store a key of 128 bits for every file in the system, regardless of its length. K_{ss} must be kept private, since there is no need to make the job of an attacker easier publishing the list of blocks.

Hash of the block, both basic and secure hash. With the same considerations with random identifiers in respect to size, hashes have the advantage that they could be used to ensure integrity of data or event prevent unauthorized overwriting, as will be stated in next sections. These two advantages are enough to make advisable to calculate the hash code of each block and store it in the metadata file.

$$H = \{hash_{K_{ss}}(B_1), \ldots, hash_{K_{ss}}(B_i)\} \tag{15.10}$$

15.3.4. Metadata

At this moment the user's node has a local buffer with the blocks of the file B, a set H to identify each block and a configuration file, as stated in section 15.3.1. Metadata is then managed in the same way than file objects. It is padded, encrypted with K_{ff} and divided in blocks using IDA. Many of the special data is stored in a special block that we call iNode, in the same sense that iNodes in the traditional filesystems. iNodes store the identification of the set K, the whole contents of H and the identifiers of the rest of metadata blocks.

In SCFS metadata blocks are published as regular blocks. Since they have the same length and entropy as any other block, they cannot be distinguished from regular blocks. Identifiers are assigned in the same way as block identifiers. The iNode is identified in a special way, described in the next section. The first block is referred with a special identifier. Metadata blocks are then randomly introduced in the local buffer. This way, an eavesdropper cannot discriminate between file blocks and metadata blocks.

Securing File Identifications

Every file in SCFS is related to a URI in an one-to-one relation. This means that an URI identifies a file and any file is identified just with one URI. Furthermore, it should not be possible to find out the URI from any number of parts or blocks of the file.

Users store in SCFS their private data with an arbitrary name, and the file system must provide with a separate name-space of files for each one of the users. In this regard, a human readable filename must be the main interface of the user to the system. The complexity and security involved in mapping that string to a DHT identifier must be hidden to the user.

If an attacker is able to identify the iNode, with convenient crypt-analysis he may be able to retrieve the complete metadata and then the complete file. In this regard, securing the identification of the first metadata block is crucial for the security of the system.

There is a private name-space for each user in SCFS. Private name-spaces have an associated secret key K_d. The filename is hashed and then encrypted using this key. The hashing step maximizes the entropy of the encrypted string. The encryption step warrants that the user-space is unique and secured, since none can identify the file even if he knows the author and the filename.

$$ID_f = hash_{K_d}(UID, filename) \tag{15.11}$$

15.3.5. Publishing the File

At this moment, the node has a local buffer with a set of blocks B that are identified with H and the iNode of the system identified with ID_f. The node publishes the contents of the local buffer in the DHT under their keys. Readers will notice that since blocks, metadata and the iNode were randomly introduced in the local buffer, an attacker is not able to put them aside and a crypt-analysis is really difficult.

15.3.6. Reading Process

The reading process is the inversion of the writing process. From a filename users create a ID_f, maybe using a user key K_d to maintain privacy. From ID_f, the user gets the iNode of the file and then the list of metadata block. From the metadata, the user is able to recreate H. Finally, the user downloads the blocks, undoes the shuffling and then decrypts the file.

15.3.7. Implementation and Additional Mechanisms

The ideas in the previous sections were implemented in a prototype accessible in [136]. The prototype uses Kademlia [88] as the algorithm for the DHT, blocks have 2048 bytes and there is a 30% of redundancy in the IDA algorithm. Identifiers of files, nodes and users have 128 bits. AES is used as the encryption algorithm, and the first 128 bits of SHA as a hashing algorithm. Kademlia was chosen as the DHT algorithm since its buckets are more stable against ring breakages than Chord, the DHT that CFS uses. The prototype is written in Python and released under the GPL license.

Apart from the mechanisms of the previous sections, our implementation of SCFS includes others to enhance the security of the system. In this section we will study these additional mechanisms than take place in phases other that reading and writing data.

Attackers may join a large number of malicious nodes in order to perform a Sybil attack. This way, the attacker gains a large influence in a region of the Kademlia ring and he may be able to put the complete system down or perform denial of service attacks to some users. In order to prevent this kind of attacks, the original designers of CFS proposed that the identifiers of the nodes must depend on the network address of the node. Furthermore, the random identifiers of each block disperse blocks in the whole ring, while the IDA warrants that the file can be retrieved even if a segment of the ring is not available. Kademlia is especially strong against breakage of the ring [88].

Attackers may collect information by means of sniffing the communications of users in the network. Files are encrypted, shuffled and broke down in blocks as explained in section 15.3.3, but metadata have a slightly weaker encoding process, specially the first of them. An attacker that gets the first iNode needs to decrypt a single block of data to get the list of every block in the system. That first iNode is encrypted using AES, but it contents known information that may simplify the crypt-analysis. An attacker may identify this first iNode because it will be the first block than a user demands. This way, SCFS asks for several random blocks apart from the iNode.

There is no deletion service in SCFS. A deletion service needs to authenticate the owner of the file in order to prevent deletion from unauthorized users. In order to enhance privacy of the users, SCFS does not include any authentication mechanisms. In order to prevent exponential growing with time, data is actually deleted in nodes if owner does not access to the block after a month. In order to show interest in a block without the need of downloading it and increasing the bandwidth, users must send a "ping" to the data block. Nodes count these pings as an access and will mark the block as no removable for an extra month.

Both malicious and fair users may overwrite blocks of legitimate users. Since identifiers of blocks are random numbers, collisions are possible. Kademlia supports storing different blocks under the same key, and when a user demands that key he will receive the whole collection. SCFS takes advantage of this by means of saving the hash of the block in the iNodes. In this regard, when a user demands a key and receives several blocks, he is able to discriminate which one is the valid to create the original file. This mechanism does not prevent that malicious users are able to write blocks under the same key, but their blocks won't be used to recreate the original file.

Directories are files with an unordered list of files or directory identifiers that it holds. Raw content of a directory is managed as any other regular file in the system. SCFS uses different URIs to distinguish between files and directories.

15.4. Denial of service

In this section, we analyze how successful a denial of service attack is in SCFS.

A malicious node does not answer to the petitions of a block from a user. The lack of response could be willing (the node does not want to answer the petition) or unwilling (the node disappeared from the network). In this section, we analyze the probability that a malicious node is successful in its attack, P_E.

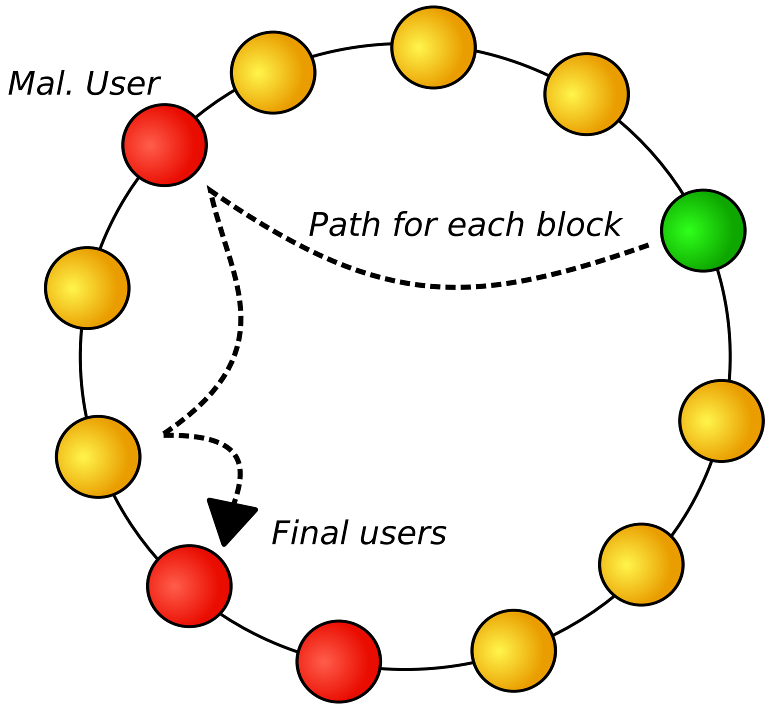


Figure 15.3.: A malicious user in the middle of the path of a block.

The probability of a successful download of any file block equals the probability of not finding any malicious node in the path between the user and the node that stores the block, as Figure 15.3 shows. That is, the probability that k nodes in the path are not malicious.

$$P_\beta(k=1) = \frac{N-M}{N} \tag{15.12}$$

$$P_\beta(k=2) = \frac{N-M}{N}\frac{N-M-1}{N} \tag{15.13}$$

$$P_\beta = \prod_{i=0}^{k-1} \frac{N-M-i}{N} \tag{15.14}$$

The user needs at least b blocks out of B to recover a whole file. If C_b^a is the binomial coefficient of a and b, we can calculate the success probability P_E as a Bernoulli experiment:

$$P_E = \sum_{j=b}^{B} C_j^B P_\beta^j (1 - P_\beta)^{B-j} \tag{15.15}$$

$$P_E = \sum_{j=b}^{B} \frac{B!}{j!(B-j)!} (\prod_{i=0}^{k-1} \frac{N-M-i}{N})^j (1 - \prod_{i=0}^{k-1} \frac{N-M-i}{N})^{B-j} \tag{15.16}$$

Figure 15.4 shows P_E for these values:

$$N = 10000 \tag{15.17}$$

$$M = 100 \tag{15.18}$$

$$k = \log 10(N) = 4 \tag{15.19}$$

$$b = \lceil 0.9B \rceil \tag{15.20}$$

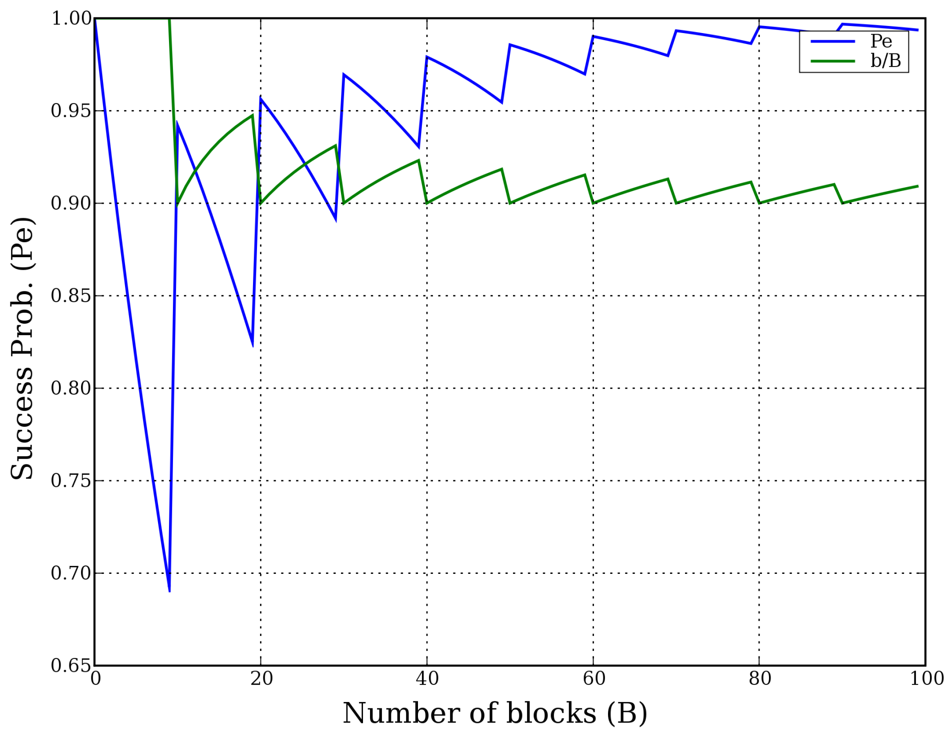


Figure 15.4.: Probability of reconstructing the original file (P_E).

Figure 15.4 shows $P_E(B)$. That functions grows "exponentially". For example, when there are $B = 50$ file blocks and $b = 45$ are enough to reconstruct the whole file, if the network has a size of $N = 10.000$ and $M/N = 1\%$ of them are malicious, the probability of recovering the complete file is over 95%. In any case, "exponential growing" of figure

15.4 is unorthodox, and there are lots of steps with differences up to 15% with a single block. Indeed, if $B = 20$ and we need $b = 18$ blocks to reconstruct the file, the successful probability goes down to 83%. Figure 15.4 simulates the system for $b = 0.9B$, and gives a hint about the reason of the large differences of the last figure: differences maximize when the quantification introduced by the integer coefficients maximizes.

In order to check how parameter $b/B \in \mathbb{Q}$ affects the output of P_E, we will perform a new test. In this case, we use the same conditions as in the last experiment, but we keep constant the number of blocks of the file $B = 100$ while modifying the number of blocks b that we need to recover it. The rest of the parameters are the same that in the last case.

$$N = 10000 \tag{15.21}$$
$$M \in \{100, 500, 1000, 2000, 5000\} \tag{15.22}$$
$$k = \log 10(N) = 4 \tag{15.23}$$
$$B = 100 \tag{15.24}$$

Figure 15.5 shows $P_E(b, M/N)$. We can observe that P_E is close to 1.0 up to a certain value for b, when it goes close to 0. Under the same conditions that in the last experiment, $M/N = 0,01$, the threshold for b is about $0.9B$, and we can explain the fast movements of figure 15.4 because we move up and down this slope. The same figure 15.5 shows P_E for several values of M/N, and the threshold for b is inversely proportional to M/N.

This threshold is an important result from the point of view of the system design because it provides an optimal for parameter b/B for each ratio of malicious nodes. We must take into account that parameter b/B is a measure of data redundancy. Optimal redundancy is exactly under the threshold, because a higher redundancy does not increases P_E. In a similar way, a redundancy close to or above the threshold decreases the system efficiency.

In the last experiments we supposed that the number of hops to reach the block is a constant $k = 4$. Obviously, this is not true. Figures 15.6 and 15.7 show the success probability P_E for different values of k, in scenarios with and without block replication. As figure 15.6 shows, the dependency of P_E with the parameter k is less than the dependency, for example, with the number of malicious users. In this regard, we conclude that the number of malicious users is more critical for the denial of service attack than the increasing of the number of hops in the network.

15.5. Security Analysis

In this section, we study some possible attacks against the system and their counter measures. These attacks are based on the objectives that were listed in 15.2.2. For a detailed comparison of SCFS with other distributed file systems, the reader is referred to [134, 135].

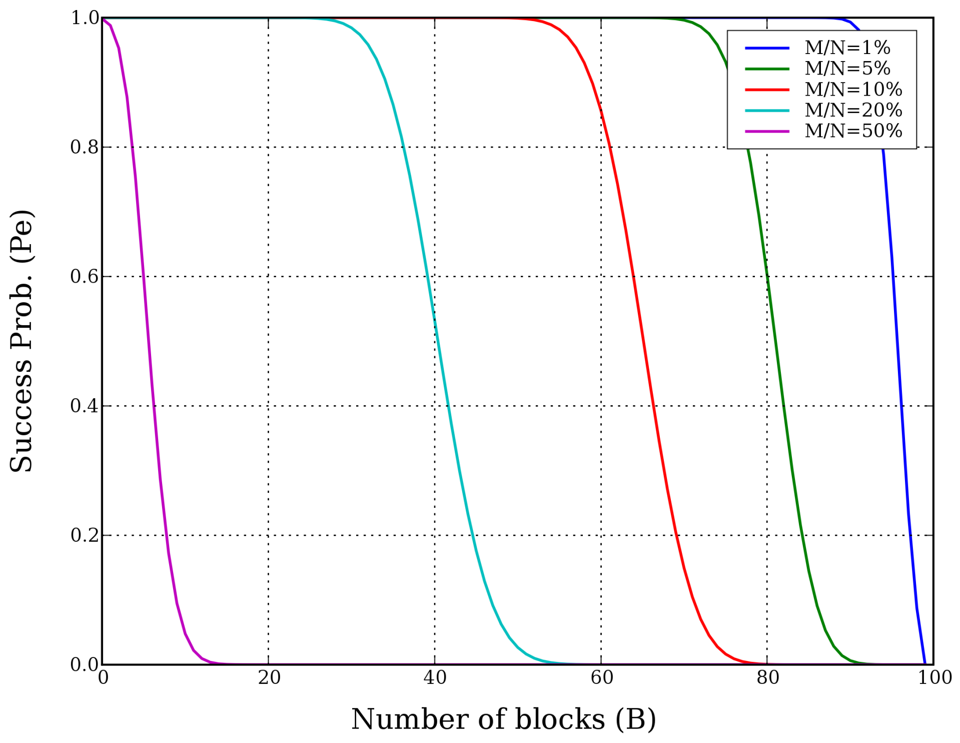


Figure 15.5.: Success Prob. as a function of M/N.

An attacker eavesdrop user's communications. Every piece of data is locally encrypted and it is never stored in plain text on SCFS. The objective of this attack is getting the root directory, the iNode or a collection of blocks of a file to perform the next attacks. SCFS asks for a number of random blocks in the first place. File blocks and metadata are indistinguishable and all of them have the same size. SCFS makes difficult for an attacker to put file blocks apart from metadata or directories.

An attacker wants to get a particular file of a user. The attacker knows of the existence of a file named in a certain way in the personal name-space of the user, as "income.xls". SCFS uses the secure hash of a filename to identify an isolated file. In this case, the attacker must break down the secure hash to get the iNode, and decrypt this block to get the list of blocks of the file. Our prototype uses random identifiers for files stored in a directory, and only the name of the directory of the root directory uses secure hash with K_d. Random identifiers are safe against this attack.

An attacker wants to list the contents of a directory Directory names are protected with K_d in the same way that filenames are. An attacker could obtain the identification of the root directory, or even its blocks, if the last attack is successful. Directories are special files that only stores a list of iNodes to their content, and thus they need a small number of blocks. It is feasible a brute force attack to rearrange blocks on these small files, but the attacker still has to break down a symmetric encryption with K_f.

An attacker gets the metadata of a file. The first blocks that a user asks to SCFS are the metadata of a file. An attacker could put apart these blocks and crypto-analyze

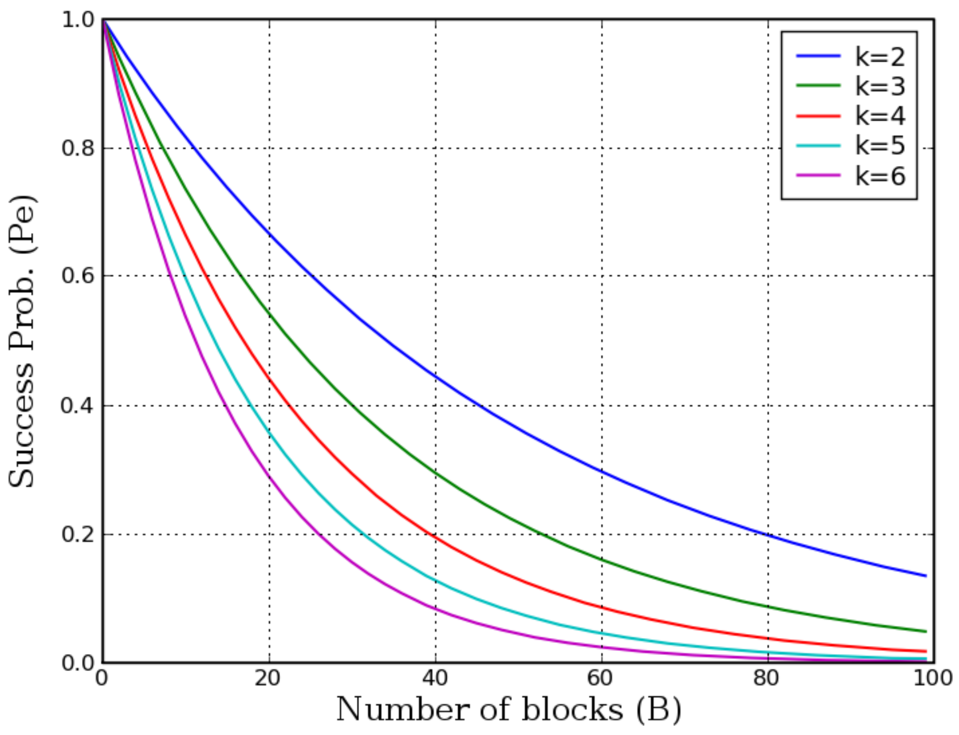


Figure 15.6.: Success Prob. as a function of k ($N = 10000, M = 100, k = 4, b = B$).

them to get K_{ff}. In this regard, SCFS asks for some random blocks apart from the actual ones to make this attack less feasible.

An attacker collects every block of a single file. If an attacker eavesdrops the communication of a user he is able to get the list of blocks that conforms a file without any need of decrypting the metadata. However, since the IDA algorithm takes place after the encryption and the attacker has no information about the K_s that creates the vector space for the algorithm, he has to permute the blocks of the file at random and then perform the cryptanalysis to break down the file encryption with K_f. The IDA step adds an additional security to the encryption that makes feasible to use a weaker algorithm for encryption of the file for constrained devices.

An attacker deletes a file of a user. Since SCFS does not offer a deletion mechanism as explained in section 15.3.7, this attack cannot be performed.

An attacker controls a group of nodes in the network. The original CFS uses Chord at the DHT level and the identifiers of each node depend on the network address. SCFS shares this behavior with CFS, and then an attacker has a limited range of identifiers for his nodes. The DHT layer is able to perform long jumps even if source and destiny are separated a long way in the ring. The buckets of Kademlia, which is the DHT used in SCFS, are even stronger against a ring breakage [88]. Furthermore, since users only need k blocks of n to reconstruct the original file, the blocks stored in the segment that the attacker controls are not really needed. As stated in [134, 135], SCFS does not currently have an economics system to detect and prevent malicious behavior. A system of this

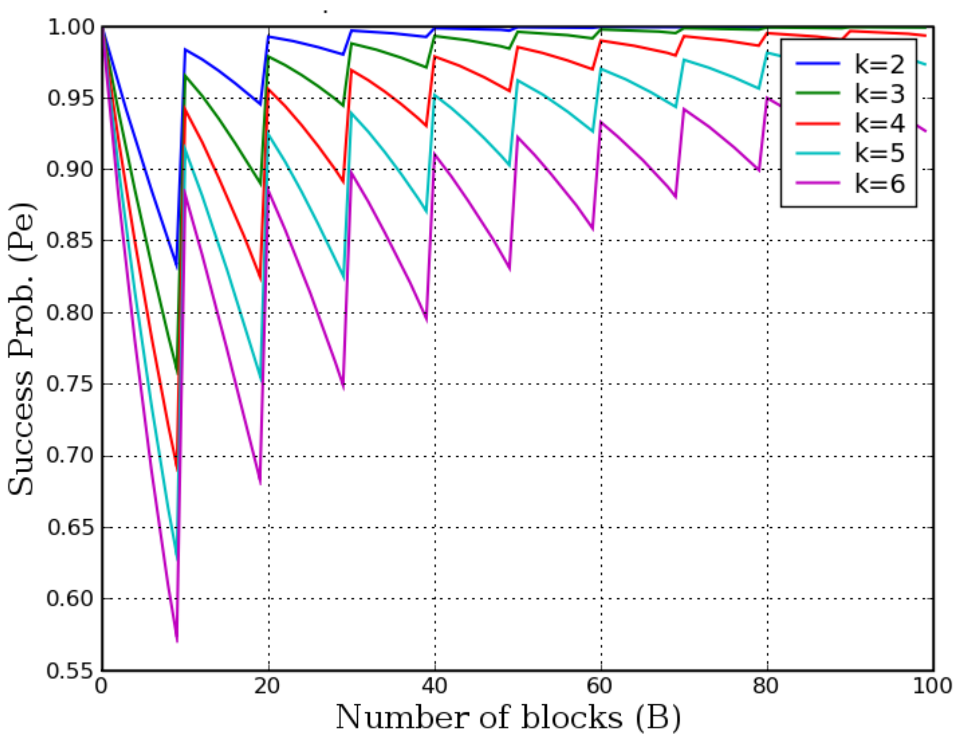


Figure 15.7.: Success Prob. as a function of k ($N = 10000, M = 100, b/B = 0.9$).

kind minimizes the dangers of malicious users in the network by means of identifying them as soon as possible.

Lazy nodes do not send a block, or an attacker sends a false block back. Nodes that are overwhelmed with links or prefer not to cooperate with other nodes won't answer to block requests. Since SCFS uses an IDA algorithm, users will only have to gather k out of n blocks to reconstruct the original file. Modification of a block is easily identified with the hash included in the iNode. An economics system to prevent such kind of attack is advisable, as stated previously.

A node try to decrypt the blocks that it stores. SCFS encrypts blocks using AES and breaks down files in pieces with the IDA. The information of a single block makes little sense to the node that stores a single block. Even the block identification contents no information about the contents, its filename or its publishers.

Any user overwrites one or more blocks of another user. Since there are many users in the network that store many files with many blocks, there is a high likelihood that a legitimate user uses the same identifier as another user for different blocks. Attackers may use the same identifier with malicious intentions. The DHT is able to store several blocks under the same identifier, and it will return the whole list after a question. Since SCFS stores the hash of the blocks in the metadata, users can discriminate their blocks from other user's block even if they are stored under the same key. Since SCFS uses SHA, we assume that an attacker has not a feasible way to modify a block while keeping the same hash and he won't be able to overwrite a user's block.

An attacker traces the publisher of a file. In this attack, we assume that the attacker could overcome the attacks listed above. That assumption needs a considerable processing power, and only major government agencies can carry out such an attack. Since there is not an authentication mechanism in the system, the agency has not any knowledge of the original author of the file. In this regard, the agency has to supplant the node that store the initial iNode and wait for the publisher to download the file to catch him. Even in this case, the network can use one of the mechanisms described in [46] or [3] to warrant access anonymity to its users.

A government agency prosecutes nodes that distribute a file. As stated in the previous attacks, nodes in SCFS do not know which kind of content they store, their contents or filename. In many countries, denying knowledge of the content that a node store is enough to prevent prosecution from local authorities to node administrators.

16. Conclusions

We defined in part I the recommender system as a complete process, which includes the access to the recommended document. Hence, the recommender system must provide a mechanism to download documents as the last step of the recommendation process.

Accessing the recommended document must be taken with care. If a user keeps his profile private, gets a recommendation using anonymous networks and plausible deniability and access the final document using a unsecured mechanism, the protection achieved during the first phases of the recommender system is useless. Indeed, an attacker may learn about the user preferences by inspection of the documents that he downloads. Then, access to the final document may expose the user and other participants of the process.

Additionally, other participants in the recommendation process are at risk. Part III protected recommenders by means of providing plausible deniability to their recommendations, but the output of the process was a $URL(d)$ pointing to a document. If recommenders can access and download documents using the URL that they index, then they can be legally prosecuted on copyright grounds. Merchants are also at risk, since if the document repository or an external attacker are able to identify the entity that published a document, they could be legally prosecuted. Regarding the documents availability, the MegaUpload case showed that it is possible that huge amounts of documents disappear after a legal attack from a foreign country.

In this part of the thesis, we analyzed several distributed filesystems from the point of view of security and concluded that, as far as we know, there was not a satisfying solution to store personal data. We analyzed the security requirements of such service and concluded that CFS was the proposal that best matches the network necessities of personal users. Then, we introduce a Secure Cooperative Filesystem that solves the security problems of CFS. The ideas behind SCFS were implemented in [136].

SCFS system works on a Distributed Hash Table using a decentralized structure. Files are divided in blocks, which are published on the DHT. Blocks include redundancy to prevent data lost, and there are some special blocks save metadata and pointers to the other blocks of the file. These special blocks are organized in name-spaces that are in turn protected using individual and/or group keys. Without the necessary key, an attacker cannot access to the main iNode of the file, and then he won't be able to reconstruct the whole file.

We analyzed the availability of documents in the distributed filesystem, and the risk of a denial of service in this filesystem. We used both analytical and simulation approaches, and arrived to a trade-off between data redundancy and the number of expected malicious users in the network.

After this analysis, we concluded that for a given value of the coefficient b/B (data redundancy in the information dispersal algorithm), there is an optimal value $\beta_{op}(M/N)$ that is related to the number of malicious nodes in the system. In addition, a system without redundancy $b/B = 1$ and a number of malicious nodes is not going to work with high probability for interesting documents $P_e(B > 50) \approx 0$. Finally, we can assume that the success P_e does not depend on the average number of hops for k for $b/B < \beta_{op}$.

As a result of the security analysis, we conclude that SCFS solves many of the security requirements of a distributed file system, but it has several possible enhancements. Particularly, we would like to do further research in order to include an economic system for SCFS.

SCFS was implemented as a real system, and published as open source.

Part VI.

Conclusions and Future Work

17. Conclusions of this thesis

In this thesis, we explored the problem of finding documents in distributed networks. We established that a recommender system was the best approach to discover new content that the user was not previously aware. In section 1.1, we defined a recommender system as an automatic system that, given a customer model and a set of available documents, is able to select those documents that are more interesting to the customer. As we explored in section 2, recommender systems are extensively used in e-commerce.

Recommender systems must show some characteristics. Getting useful recommendations fast is a desirable characteristic of a successful recommender system. Furthermore, most recommender systems use profiles to describe users and documents, and these profiles may contain personal information that must be protected. In most commercial recommender systems, user profiles are stored in a centralized server that users must trust, or are spread in a decentralized network of uncontrolled nodes. In addition, the source of a recommendation, the final provider of the resource or the intermediate nodes in the network may face charges of copyright infringements, as we found in recent times. We believed that a *better* recommender system must face all these challenges. Unfortunately, this is not the case of the most successful recommender systems in use today.

During this thesis, we explored the challenges that a recommender system must face. At the beginning of chapter 1, we established that the desirable characteristics of a recommender system are (i) to be fast, (ii) distributed and (iii) secure. A *fast* recommender system enhances the shopping experience of the client: a recommendation is not useful if it arrives too late, and the distributed and highly dynamic environment that is under our study makes fast recommendations a especially difficult task. A *distributed* recommender system makes difficult for an attacker to collect large sets of user's profiles, and prevent the creation of centralized databases with sensitive information. Finally, a *secure* recommender system protects every participant of the system: users, content providers, recommenders and intermediate nodes.

From the point of view of security, there are two main issues that recommender systems must face: (i) protection of the users' privacy and (ii) protection of other participants of the recommendation process. Recommenders issue personalized recommendations taking into account not only the profile of the documents, but also the private information that customers send to the recommender. Hence, the users' profiles include personal and highly sensitive information, such as their likes and dislikes. In order to have a really useful recommender system and improve its efficiency, we believe that users shouldn't be afraid of stating their preferences. As a consequence, the personal information that is included in the user profiles must be protected.

The second challenge from the point of view of security involves the protection against a new kind of attack. Prevention of illegal distribution of copyrighted documents by means of pure technical solutions has not been effective. Copyright holders shifted their targets to attack the document providers and any other participant that aids in the process of distributing documents, even unknowingly. In addition, new legislation trends such as ACTA or SOPA show the interest of states all over the world to control and prosecute these intermediate nodes. During section 3.1.2, we learned that recommender systems are treated by this new kind of attacks, and we concluded that recommender systems should be protected against them.

To achieve these goals, during this thesis we proposed the next contributions:

1. A social model that captures user's interests into the users' profiles, and a metric function that calculates the similarity and affinity between users, queries and documents.

2. Two mechanisms to protect the personal information that the user profiles contain.

3. A distributed system on a cloud that protects merchants, customers and indexers against legal attacks, by means of providing plausible deniability and oblivious routing to all the participants of the system.

4. A social, P2P network where users link together according to their similarity, and provide recommendations to other users in their neighborhood.

5. Mechanisms to enhance the time to create the social network, and improve its efficiency.

6. A document distribution system that provides the recommended documents at the end of the process.

We divided this thesis document in four main parts: (i) *social model*, focused on contributions 1 and 2; (ii) *DocCloud*, focused on contribution 3, (iii) *Clusters*, focused on contributions 4 and 5; and (iv) *Document distribution*, which proposed contribution 6.

Social model. The recommender system that we developed in this thesis uses social networks where users link to each other according to the affinity of their profiles. Thus, checking if two profiles are or not affine is going to be a mandatory step of such recommender system. The social model to be used in the recommender scenario was studied in part II of this thesis.

In chapter 5, we modeled users, documents and queries as vectors of a social space. In addition, we explored how similarity can be defined in an objective way, and established the definition of affinity for users, documents and queries. The user can configure the degree of affinity by means of a parameter λ that can be personalized independently. We used the expression document/query/user profile to refer to this model.

User profiles include their likes and dislikes, and these are very sensitive data that must be protected. However, the recommender needs access to these data to decide whether or not a query is affine to a document, and users of a social network need to be able to check whether they are affine. Therefore, the protection of the users' privacy should not avoid the calculation of the affinity of two profiles.

We followed two different approaches to solve this problem:

- Setting a limit to the amount of information that a profile contains about a user or document.
- Solving the question of whether two profiles are affine or not, without leaking any additional information about the profiles.

In chapter 6, we explored the first case using a distortion mechanism to protect the users' privacy during the first step of the recommendation process. We took advantage of the lemma 6.1.2 to linearly project profiles into social spaces of less dimensions. According to this lemma, the attacker is not able to separate the vector elements if the dimension of the projected space is less than half the dimension of the original social space.

But even if lemma 6.1.2 proves that the projections meet the privacy requirements of our system, it gives no clue about the usefulness of the projection. That is, whether projected vectors can be used to check the affinity of the users. We found that the projection matrix must show certain characteristics. Lemma 6.1.1 assures that there is some projection matrix that, with bounded probability, keeps distances after projections. Then, our problem is equivalent to finding a projection matrix of less than half the dimensions of the original social space.

We explored three different projection matrices and how they protect the information of the user. We defined the parameter *uncertainty* to measure the amount of information that the projected profile still includes about the original profile. In this case, we established that distances (hence, affinities) were the data to protect.

As we showed, projections create a trade-off between utility and privacy: projections into a lower space are more secure yet less usable to match profiles. In addition, triangulation attacks can be used to trace down the position of the user in the original social space, to a limit. By means of modifying the projection matrix, the user is able to change the uncertainty of the projection and then increase the privacy of the exchange, at the cost of an increasing of the number of false positives and negatives in the affinity match process.

In chapter 7, we explored a different approach. In this chapter, we introduced a zero-knowledge protocol that only answers the question of whether or not two profiles are affine, without leaking any information about the original profiles if they are not affine. Simulations showed that we were partially successful, since the protocol can be configured to bound the number of false negatives as much as necessary, but the overall number or errors increases. Despite this fact, we analyzed that in our scenario a big number of false positives is acceptable if the number of false negatives remains small. Through a theoretical analysis, we proposed suitable values for the configuration parameters γ according to the desired affinity threshold λ.

DocCloud. Next, we defined a system that protects intermediate nodes, document indexers and recommenders from legal attacks. We referred to this system as DocCloud, and it was introduced in part III of this thesis document.

We identified *plausible deniability* as the security service that recommender systems must provide in order to offer protection against legal attacks to the participants of the recommender system. We defined plausible deniability as the ability that a node has to deny any knowledge of the document that they are recommending to the users of the system. Absolute deniability is not feasible in our system, and we use a probabilistic approach to provide this service.

In order to provide plausible deniability in our system, we use three different methods.

- The privacy protection mechanisms that were introduced in part II of the thesis. Profiles are distorted using a group key that is not directly available to indexers. In the event that indexers learn this group key, the distortion cannot be undone. Furthermore, the publishers of the profiles into the indexers are behind an anonymous network and cannot be identified. As proved in part 6, profiles can be matched and affinities computed (with a bounded error) even if profiles are distorted

- A Private Block Retrieval scheme that connects customers and recommenders in such a way that recommenders cannot identify the content that they are recommending.

- A tree-shaped organization of recommenders as nodes of a cloud system to prevent the identification of the source of the recommendation, and provide plausible deniability to databases.

In chapter 10, we introduced this organization and build the recommender system using a tree of databases that run a Private Block Retrieval scheme. This way, a customer can get a recommendation from a set of recommenders without identifying the specific indexer that stored the content. In addition, the databases are oblivious to the fact of whether or not they answered the query of the user. That is to say, none of the participants of the exchange, not even databases, is able to identify the source of the recommendation.

As we showed, this structure needs a careful distribution of documents inside the databases to provide the plausible deniability service. We explored two possible distributions of data inside the tree of recommenders, a uniform distribution and a distribution of resources more similar to the one expected in a social network, as the one that the rest of the thesis create. In section 10.3.1, we provided a limit on the number of items that indexers must answer in order to provide optimal anonymity inside the indexers tree. To calculate analytically this limit, we made the assumptions of a uniform distribution of resources, and databases structures such as a social network.

We believe that these mechanisms protect indexers against legal prosecution, since: (i) they cannot identify the original publishers of document profiles; (ii) they can't undo the distortion and then learn the exact profile that they are indexing; (iii) they cannot identify the customer of the system and (iii) they don't know whether they answered the query of the customer.

Clusters In part I, we justified the use of a distributed network to organize users without any central node. In part III, we justified the necessity of organizing users as a distributed social network, where nodes link each other according to their affinities. We believe that the social network, at first a requirement of the security mechanisms, can be used to improve the results of the recommendation process. Indeed, the fact that the nodes in a social network link to other nodes that are controlled by similar users enhances the results of the recommendation process and aids in the location of new and interesting documents.

Part IV explored the fast and automatic creation of a social network where users link to each other according to their affinity. We analyzed how epidemic algorithms can be used to identify interesting documents in a non-structured P2P network where links depend on the users' preferences. In chapter 13, we described an epidemic routing algorithm that creates links according to the affinities of the network.

We found that affinities alone are not enough to find new similar users. Apart from affinities, we set two additional criteria to create links in the social network: (i) clustering and (ii) random links. The use of these two criteria tries to induce the behavior of a small world network.

Again, we showed that these new criteria alone do not improve enough the results of the epidemic algorithm. Then, we proposed a SoftDHT, a structure of sample user profiles that aids in the location of islands of similar users that were not identified at first. In addition, we discussed that there is little gain if a node links to another node that already belongs to a highly clustered affinity group. It may be more useful to limit the number of neighbors and include links to other less clustered, external nodes. In this case, external and unknown groups of users that share the same interests but have not being discovered yet can be found.

These ideas and improvements were tested in section 13.4. We performed throughout simulations, and we found that the proposed enhancements improve the performance of basic epidemics algorithms in dynamic scenarios and shortens the convergence time, while having a comparable performance in the long run. In addition, we tested the network structure of the basic searching algorithm and the improved version, and found that the improvements aid in the creation of a network that shows the desired small world behavior.

Document distribution As established in part I, we view the recommender system as a complete process, which includes the access to the recommended document.

Thus, a convenient mechanism to access and download the recommended documents must be provided. This is a key difference with many other recommender systems in the literature. Most of them considered that the recommender process ends when the user gets a recommendation, and do not take into account how the customer gets the recommended document. We believe this is an error for several reasons:

- Regardless how protected the recommendation process is, accessing to the final document may expose the user and other participants of the network at risk. If users are not protected during this step, the kind of document that they retrieve tells as much as their interests. This is more evident if the attacker analyzes the behavior of the user in the long term. Then, we believe that it is useless to protect the user during the recommendation phase if the document download is not protected as well.

- Recommenders may be at risk. Part III protected recommenders by means of providing plausible deniability to their recommendations, but the output of the process was a $URL(d)$ pointing to a document. If they output a recommendation of a copyrighted document pointing to a document that they can access and download, they may be charged on copyright grounds.
- Merchants are at risk. If the document repository or an external attacker is able to identify the entity that published a document, they could be legally prosecuted.
- Documents may disappear from the system. Cases such as MegaUpload showed that it is possible that large, centralized repositories were taken down by foreign authorities on copyright grounds. This way, a decentralized filesystem improves the document availability and makes the system more secure against legal attacks.

During part V, we analyzed a distributed and secure filesystem that copes with these issues. We called this system the Secure Cooperative Filesystem (SCFS). SCFS system works on a Distributed Hash Table, using a decentralized structure. The distributed filesystem works by splitting the original file in blocks and saving them in the DHT. Blocks include redundancy to prevent data lost. In addition, some special blocks save metadata and pointers to the other blocks of the file. These special blocks are organized in name-spaces that are in turn protected using individual and/or group keys. Without the necessary key, an attacker cannot access to the main iNode of the file, and then he won't be able to reconstruct the whole file.

SCFS was implemented as a real system, and published as open source.

The system described in chapter 15 is able to provide (i) confidentiality, in the sense that only users with the suitable key are able to access the document; (ii) privacy, since attackers cannot access any piece of data or metadata from the file or any of its blocks; (iii) integrity and persistence, preventing data lost and modification of the files; (iv) availability, by means of providing mechanisms that allow restoring the original file even if some of its blocks are missing.

Finally, we analyzed the availability of documents in the distributed filesystem, and the risk of a denial of service in this filesystem. We use both analytical and simulation approaches, and arrived to a trade-off between data redundancy and the number of expected malicious users in the network. In section 15.4, we showed this trade-off and the suitable value for the configuration parameter for different scenarios.

17.1. Adequacy to the initial objectives

The objectives of this thesis document were established in section 3.2. As the reader will recall, we aimed to the creation of a recommender system that was (i) hybrid, from the point of view of the input data that the recommender uses; (ii) distributed; (iii) a complete process; (iv) with separate actors for roles, for improving the security of each participant; and (v) secure, especially from the point of view of preventing the exposure of private data and protection of all the participants in the system.

In part II of this thesis, we constructed a semantic model that captures the information of both users and documents, assigning to them vectors of a social space. Additionally, during the development of this thesis we have used several metrics in order to calculate the similarity and affinity of users, documents and queries. Since the inputs of these metrics and models are the profiles of documents and users, that is, the description of the user and the categorization of documents according to their contents, the recommender system that we have developed is a hybrid recommender system.

The main actors of our recommender systems are distributed in several networks. Recommenders and document indexers construct a cloud computing network to protect them against legal attacks; customers organize a dynamic social network that links together affine users to enhance the results of the recommendation process; documents are distributed from a filesystem that is organized as a secure DHT. Therefore, each of the different parts of our recommender process is distributed, and there is not any central node in our system. This thesis has built a distributed recommender system.

During part V, we described a distributed filesystem. This SCFS is used to store documents and provide security to merchants and customers. We believe that providing access to the final document, and keeping this access secure, is a mandatory step of a recommender system. The SCFS meets this challenge.

The security of the different participants has been faced according to their roles. The DocCloud that was described in part III of this document separates the participants in roles, and defines different security mechanisms for each role. Indexers were protected using the plausible deniability service, and customers were protected by means of the privacy mechanisms proposed in part II. Merchants and document providers were protected using the distributed filesystem of part V. Hence, all participants of the recommender system are protected according to their roles. The reader will recall that, being a decentralized recommender system, all participants may play different roles at different moments.

As a consequence, we can conclude that we have meet the objectives sets during part I of this thesis.

18. Future Work

In this chapter we analyze the open issues and open lines that our recommender system presents. For the sake of simplicity, we will use the same structure in parts that we used in the rest of the thesis document.

Social model. Accurate metric and profile calculations have not being faced in this thesis. We defined a simple user and document model, and we only analyzed the effect of the cosine metric on the privacy of the users. In this regard, the user model should be extended to cope with the more modern models, and additional metrics should be evaluated.

Regarding the zero knowledge protocol of chapter 7, there is room for improvements. We considered that the social space is clustered with hypercubes. Parameter γ should be the average diameter of a cluster, and hypercubes have so much variance in the length of their diameters that it may unnecessarily increase the errors that the protocol outcomes. In this regard, we must consider how the social space should be clustered to create optimal partitions and improve the true positive rate. Furthermore, we believe that it is possible to dynamically define the common input G and clustering function c in such a way that false positives are minimized and prevent sophisticated attacks to control some areas of the social space. These are open lines for future works.

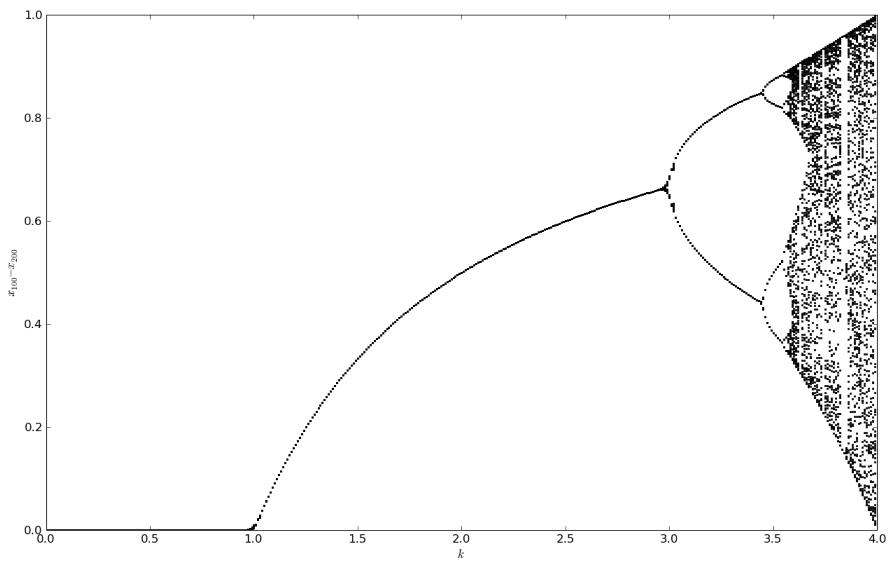
We are excited with an additional approach to protect the privacy of the user profiles that makes use of the logistic map. Given a seed $0 < x_0 < 1$ that is randomly chosen and a constant $k > 0$, the logistic sequence is created by iteration of the logistic map.

$$x_i \to \begin{cases} U(0,1) & \text{if } i = 0 \\ kx_{i-1}(1 - x_{i-1}) & \text{otherwise} \end{cases}$$

After some iteration, this sequence may converge to a stable fixed point, oscillate between unstable fixed points or have no stable behavior. The specific behavior of the sequence depends only on the value of the parameter k. Figure 18.1 shows an example of the values of x_{100} to x_{200} for $k \in [0, 4]$. In each case, the value of the initial seed x_0 was chosen at random. This graph is called the bifurcation diagram of the logistic sequence.

We believe that, with the the use of secure multi-party calculation schemes, we can use the logistic map or any other sequence with a chaotic behavior to improve the privacy of the user. In this regard, two affine profiles will use the stability region of the logistic sequence, while not affine profiles will end inside the chaotic region.

Figure 18.1.: Fig-tree from $k = 0$ to $k = 4$ for x_{100} to x_{200}



DocCloud. There are some open lines of improvement in this system. When a user gets the *URL* of an interesting document, she still has to contact another network to finally download the document. It is not clear whether or not the process of downloading can be separated from the process of selecting documents. For example, an attacker controls an indexer that forges special *URLs* in such a way that he is able to decide whether or not they are downloaded afterward. This way, he would be able to link a query to a user. This kind of attack was not addressed in this document.

A second open line of research is the management of the social cloud. If cluster A is created only with users with similar profiles, then a "representative" profile may be calculated for the cluster, and it may be close enough to the individual descriptions of each user to unacceptably leak private information that can be used to learn the users' profiles. In this case, we devise that clusters should be created with users with several "classes" of profiles. The impact of these "multi-ethnic" clusters on the efficiency of the system remains unclear.

Additionally, during this work we described a theoretical model to protect document indexers of a cloud system. We are testing the system and its proposed mechanisms using a real implementation in order to have an idea of the impact that they have on the quality of the recommendations that users obtain from the system in large social networks. This approach is obviously limited to the quality of the assumptions that we force upon the user descriptions. An implementation on an actual social cloud will be the real test for this proposal. We are currently working on this implementation.

Streaming documents In this thesis, we explored a document distribution system that gets documents from a distributed, static filesystem. While distributed filesystems are useful for small size files and off-line view of large files, it is not the best solution for large and sequential documents such as movies, music or TV channels.

The adaptation of the ideas of this thesis to a streaming service is one of our future work lines. Some preliminary results in this line were presented in [133]. The first tests on a private streaming service showed that even with a single document provider with many file entries, it was possible to serve a stream of as much as 10Mbps from a single node. The first approach used selection vectors in the same sense that 10.2 with thousands of possible selections. Even a large computer needs the order of minutes to encrypt the selection vector of several thousand entries, and the same amount of time to decrypt the resulting stream. A clever buffering policy of encryptions may improve speed at the run time at the expense of a extremely slow creation of these initial values. The streaming service was analyzed using smart-phones as final points of the network, and this problem is much more evident on these devices due to its constraints and processing power.

In [133], we structured documents (audio files and questions, in that context) as a tree, as figure 18.2 shows. For each level, we provide k databases. Each database stores an item of each one of the branches. In this way, when a user downloads a specific question of a level, he first identifies the database that stores the item, and uses a PBR scheme similar to the one presented in part III to obfuscate the branch that is following at this moment.

The structure that is presented in this section needs k databases in each level of the questions' tree, and a total of kL databases. This structure of databases should be organized as a cloud managed by the MSRP server. The challenges to solve in the streameable approach are the size of the selection vector, and the limited resources of the targeted devices.

Despite our initial simulations and analysis, the final system as a whole was not implemented. We are especially concerned about the creation of the selection vector in a smart-phone with constrained memory and processing power. Even if we feel that the results provided by the theoretical analysis are promising, an implementation of the final system to validate our ideas is very convenient. Currently, we are working on this implementation.

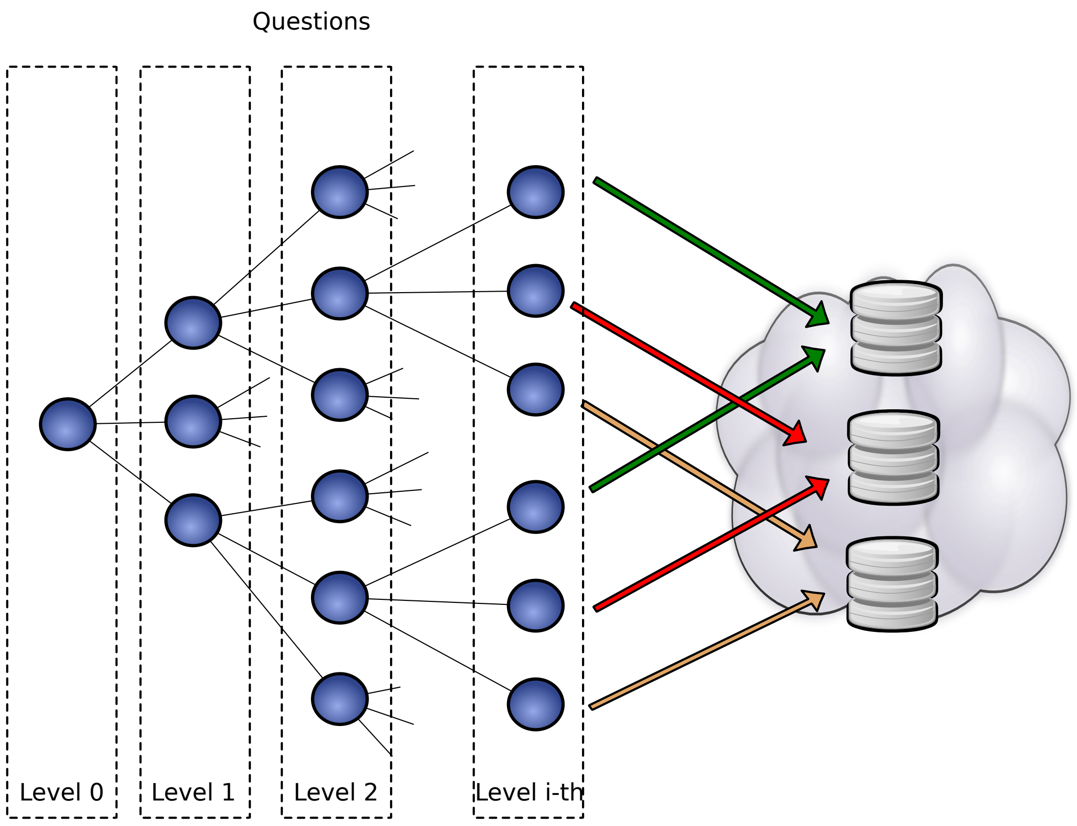


Figure 18.2.: The tree of questions and databases

19. List of publications

International Journals

- “Low-cost group re-keying for unattended wireless sensor networks”, Juan Hernández Serrano, Juan Vera del Campo, Josep Pegueroles and Carlos Gañán. Wireless Networks. 2012. Accepted.
- “Design of a P2P Content Recommendation System using Affinity Networks”, Juan Vera del Campo, Josep Pegueroles, Juan Hernández Serrano and Miguel Soriano. Computer Communications. 2012. Accepted.
- “DocCloud: a Document Recommendation System on Cloud Computing with Plausible Deniability”. Juan Vera del Campo, Josep Pegueroles, Juan Hernández-Serrano and Miguel Soriano. Information Science, 2012. Major changes.
- “Providing Security Services in a Resource Discovery System”, Juan Vera-del-Campo, Josep Pegueroles, and M. Soriano. Journal of Networks. Academy Publishers, no. 2, pp. 1, 2, 2007, ISSN: 1796-2056.

International Conferences

- “Providing Security Services in a Multiprotocol Service Discovery System for Ubiquitous Networks” Juan Vera del Campo, Josep Pegueroles, Miguel Soriano. First Conference on Availability, Reliability and Security (ARES)
- “Demonstration of Security in Service Discovery and Access for Ubiquitous”. Juan Vera del Campo, Juan Hernández and Josep Pegueroles. 25th Conference on Computer Communications IEEE INFOCOM 2006, 2006.
- “Seguridad en Protocolos de Descubrimiento de Servicios de Redes Heterogéneas”, IX Reunión Española sobre Criptología y Seguridad de la Información, 2006
- “SCFS: Towards Design and Implementation of a Secure Distributed Filesystem”. Juan Vera del Campo, Juan Hernández and Josep Pegueroles. International Conference on Security and Cryptography (SECRYPT), 07/2008.
- “Profile-Based Searches on P2P Social Networks”. Juan Vera del Campo, Juan Hernández and Josep Pegueroles.Ninth International Conference on Networks (ICN), 2010 on, pp. 98 -103, 04/2010

- "Private Audio Streaming for an Automated Phone Assistance System". Juan Vera del Campo, Ana Gómez Muro and Miguel Soriano 2011. International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pp. 128-133, 10/2011, ISBN: 978-1-4577-1448-1, DOI: 10.1109/3PGCIC.2011.62

National Conferences

- "Diseño seguro de una plataforma de e-gobierno". Joan Tomàs-Buliart, Juan Vera-del-Campo, Miguel Soriano and Josep Pegueroles. VI Jornadas de Ingeniería Telemática, 09, 2007
- "Análisis de Seguridad de un Sistema de Archivos Distribuido" Juan Vera del Campo, Juan Hernández and Josep Pegueroles. X Reunión Espanola sobre Criptología y Seguridad de la Información (RECSI'08), 09/2008
- "Búsquedas epidémicas basadas en perfiles en redes P2P no estructuradas", Juan Vera del Campo, Juan Hernández and Josep Pegueroles. VIII Jornadas de Ingeniería Telemática 2009
- "Comparación de afinidades privada mediante isomorfismo de grafos" Juan Vera del Campo, Juan Hernández and Josep Pegueroles. Tarragona (Spain), XI Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2010), 09/2010.

Bibliography

[1] MNet-Mojo Nation. Web page: mnetproject.org.

[2] StyleHop. http://www.crunchbase.com/company/stylehop.

[3] TOR Anonymity Online. Webpage: http://www.torproject.org.

[4] FastTrack, Peer-to-Peer Technology Company. www.fasttrack.nu, 2001.

[5] *Hybrid Search Schemes for Unstructured Peer-to-Peer Networks*, 2005.

[6] Abmahnung für Uploader dank Zivilrechtlichem Auskunftsanspruch. Web: http://www.gulli.com/news/rapidshare-abmahnung-f-r-2009-04-25/, April 2009.

[7] FimAffinity / MovieAffinity. FilmAffinity. http://www.filmaffinity.com/en/main.html, (Visited March, 2012).

[8] D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 2003.

[9] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, june 2005.

[10] Gediminas Adomavicius and Alexander Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17:734–749, 2005.

[11] Aggregate Knowledge, Inc. Aggregate Knowledge. http://www.aggregateknowledge.com/, (visited July, 2012).

[12] Amazon Company. Amazon: Online shopping for electronics, apparel, computers, books, dvds & more. www.amazon.com, (visited July, 2012).

[13] Mauro Andreolini and Riccardo Lancellotti. A flexible and robust lookup algorithm for p2p systems. *Parallel and Distributed Processing Symposium, International*, 0:1–8, 2009.

[14] Amelie Anglade, Marco Tiemann, and Fabio Vignoli. Complex-network theoretic clustering for identifying groups of similar listeners in p2p systems. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 41–48, New York, NY, USA, 2007. ACM.

[15] Siddhartha Annapureddy, Michael J. Freedman, and David Mazières. Shark: Scaling File Servers via Cooperative Caching. *NSDI*, 2005.

[16] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore Art Thou R3579X? Anonymized Social Networks, Hidden Patterns, and Structural Steganography. In *International World Wide Web Conference Comittee*, May 8-12 2007.

[17] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*, chapter 3, page 75. ACM Press, 1999.

[18] Xiuguo Bao, Binxing Fang, and Mingzeng Hu. Cocktail search in unstructured P2P networks. *Grid and Cooperative Computing - GCC 2004 Workshops, GCC 2004 International Workshops IGKG, SGT, GISS, AAC-GEVO, and VVS. Proceedings (Lecture Notes in Computer Science)*, 3252:286–93, 2004.

[19] M. Bawa, R. Jr, S. Rajagopalan, and E. Shekita. Make it Fresh, Make it Quick – Searching a Network of Personal Webservers. In *12th Int. World Wide Web Conference*, 2003.

[20] Sebastian Kay Belle and Marcel Waldvogel. Consistent Deniable Lying: Privacy in Mobile Social Networks. In *Workshop on Security and Privacy Issues in Mobile Phone Use*, 2008.

[21] N. Bisnik and A. Abouzeid. Modeling and analysis of random walk search algorithms in P2P networks. In *Second International Workshop on Hot Topics in Peer-to-Peer Systems*, pages 95–103, San Diego, CA, USA, 2005.

[22] J.S. Breese, D. Heckerman, C. Kadie, et al. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[23] John Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 238–245, New York, NY, USA, 2002. ACM.

[24] Kyle Chard, Simon Caton, Omer Rana, and Kris Bubendorfer. Social Cloud: Cloud Computing in Social Networks. In *Proceedings of the 3rd International Conference on Cloud Computing IEEE Cloud 2010*, 2010.

[25] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like P2P systems scalable. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 407–418, New York, NY, USA, 2003. ACM Press.

[26] Paul Alexandru Chirita, Andrei Damian, Wolfgang Nejdl, and Wolf Siberski. Search Strategies for Scientfic Collaboration Networks. In *P2P-IR*, 2005.

[27] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009/2001 of *Lecture Notes in Computer Science*, page 46. Springer Berlin / Heidelberg, July 2000.

[28] Christopher W. Clifton and Mummoorthy Murugesan. Providing privacy through plausibly deniable search. April 2009.

[29] Francesc Comellas and Michael Sampels. Deterministic small-world networks. *Physica A: Statistical Mechanics and its Applications*, 309(1-2):231–235, 2002. Small-world networks;.

[30] Luigi P. Cordella, Pasqualle Foggia, Carlo Sansone, and Mario Vento. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1367, 2004.

[31] Tom Crecelius, Mouna Kacimi, Sebastian Michel, Thomas Neumann, Josiane Xavier Parreira, Ralf Schenkel, and Gerhard Weikum. Making sense: socially enhanced search and exploration. *Proc. VLDB Endow.*, 1(2):1480–1483, 2008.

[32] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 202–215, New York, NY, USA, 2001. ACM Press.

[33] "Juzgado de lo Mercantil No 4 de Madrid". Sentencia 00244/2011. http://estaticos.elmundo.es/documentos/2011/12/19/sentencia_pablo_soto.pdf, December 2011.

[34] Andreas Deutsch, Niloy Ganguly, Tore Urnes, Geoffrey Canright, and Márk Jelasity. Implementation for Advanced Services in AHN, P2P Networks. Technical report, Università di Bologna, January 2005. BISON Project, IST-2001-38923.

[35] C. Diaz. Anonymity metrics revisited. In *Dagstuhl Seminar on Anonymous Communication and its Applications, Dagstuhl, Germany*, 2005.

[36] Claudia Díaz, Carmela Troncoso, and Andrei Serjantov. On the impact of social network profiling on anonymity. In *PETS '08: Proceedings of the 8th international symposium on Privacy Enhancing Technologies*, pages 44–62, Berlin, Heidelberg, 2008. Springer-Verlag.

[37] Roger Dingledine, Michael J. Freedman, and David Molnar. The Free Haven Project: Distributed Anonymous Storage Service. *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 2000, Proceedings:*, 2009/2001:67, June 2001.

[38] J. Domingo-Ferrer, F. Sebé, and J. Castella-Roca. On the security of noise addition for privacy in statistical databases. In *Privacy in statistical databases*, pages 519–519. Springer, 2004.

[39] Josep Domingo-Ferrer, Maria Bras-Amoros, Qianhong Wu, and Jesus Manjon. User-private information retrieval based on a peer-to-peer community. *Data & Knowledge Engineering*, 68(11):1237–1252, 2009. Including Special Section: Conference on Privacy in Statistical Databases (PSD 2008) - Six selected and extended papers on Database Privacy.

[40] Niels Drost, Elth Ogston, Rob V. van Nieuwpoort, and Henri E. Bal. ARRG: real-world gossiping. In *16th international symposium on High performance distributed computing*, pages 147–158, New York, NY, USA, 2007. ACM.

[41] O.H. Embarak. A method for solving the cold start problem in recommendation systems. In *Innovations in Information Technology (IIT), 2011 International Conference on*, pages 238–243. IEEE, 2011.

[42] enigmax. Cyberlocker ecosystem shocked as big players take drastic action. http://torrentfreak.com/cyberlocker-ecosystem-shocked-as-big-players-take-drastic-action-120123/, January 2012.

[43] Ernesto. Download a copy of the pirate bay, it's only 90 mb. http://torrentfreak.com/download-a-copy-of-the-pirate-bay-its-only-90-mb-120209/, February 2012.

[44] Patrick Euster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Maussoulie. From epidemics to distributed computing. *IEEE Computer*, 37(5):60–67, May 2004.

[45] M. Fleischman and E. Hovy. Recommendations without user preferences: A natural language processing approach. In *Intelligent User Interfaces (IUI)*, 2003.

[46] Michael J. Freedman and Robert Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *CCS*, 2002.

[47] Craig Gentry and Zulfikar Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate. In *Automata, Languages and Programming*, volume 3580/2005 of *Lecture Notes in Computer Science*, pages 803–815, 2005.

[48] William T. Glaser, Timothy B. Westergren, Jeffrey P. Stearns, and Jonathan M. Kraft. Consumer item matching method and system. Patent number: 7003515, 2002.

[49] Andrea Glorioso, Ugo Pagallo, and Giancarlo Ruffo. The social impact of p2p systems. In Xuemin Shen, Heather Yu, John Buford, and Mursalin Akon, editors, *Handbook of Peer-to-Peer Networking*, pages 47–70. Springer US, 2010. 10.1007/978-0-387-09751-0_2.

[50] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[51] Songjie Gong. A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering. *Journal of Software*, 5(7):745–752, July 2010.

[52] Google Inc. Google. www.google.com, (visited July, 2012).

[53] Dima Grigoriev and Vladimir Shpilrain. ZERO-KNOWLEDGE AUTHENTICATION SCHEMES FROM ACTIONS ON GRAPHS, GROUPS, OR RINGS. 2008.

[54] Christian Grothoff, Krista Grothoff, Tzvetan Horozov, and Jussi T. Lindgren. An Encoding for Censorship-Resistant Sharing. http://gnunet.org/, 2006.

[55] GroupLens Reserch at the University of Minnesota. MovieLens. http://www.movielens.org/login, (visited on March, 2012).

[56] Daniel L. Guidoni, Raquel A. F. Mini, and Antonio A. F. Loureiro. On the Design of Resilient Heterogeneous Wireless Sensor Networks based on Small World Concepts. *Computer Networks*, In Press, Accepted Manuscript:–, 2009.

[57] Ragib Hasan, Zahid Anwar, William Yurcik, Larry Brumbaugh, and Roy Campbell. A Survey of Peer-to-Peer Storage Techniques for Distributed File Systems. In *IEEE International Conference on Information Technology (ITCC)*, April 2005. Las Vegas.

[58] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[59] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[60] O. Hermoni, N. Gilboa, E. Felstaine, and S. Shitrit. Deniability — an alibi for users in p2p networks. pages 310–317, Jan. 2008.

[61] J. Hernández-Serrano, J. Vera del Campo, J. Pegueroles, and C. Gañán. Low-cost group rekeying for unattended wireless sensor networks. *Wireless Networks*, page 1–21, 05/2012 2012.

[62] Parker Higgins. What's on the blacklist? three sites that sopa could put at risk. https://www.eff.org/deeplinks/2011/11/whats-blacklist-three-sites-sopa-could-put-risk, November 2011.

[63] D. Hitz, J. Lau, and M. Malcolm. File system design for an nfs file server appliance. In *Proceedings of the USENIX Winter 1994 Technical Conference*, pages 235–246. San Francisco, CA, 1994.

[64] J.H. Howard and Carnegie-Mellon University. Information Technology Center. *An overview of the andrew file system*. Carnegie Mellon University, Information Technology Center, 1988.

[65] Wenjin Hu, Tao Yang, and Jeanna N. Matthews. The good, the bad and the ugly of costumer cloud storage.

[66] Z. Huang, W. Chung, and H. Chen. A graph model for E-commerce recommender systems. *Journal of the American Society for information science and technology*, 55(3):259–274, 2004.

[67] IMDb.com, Inc. An Amazon.com Company. Internet Movie Database. www.imdb.com, (visited July, 2012).

[68] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems An Introduction*. Cambridge University Press, 2011.

[69] Edwin Thompson Jaynes. *Probability Theory: The Logic of Science: Principles and Elementary Applications*. Cambridge University Press, 2003.

[70] Thorsten Joachims, Dayne Freitag, and Tom Mitchell. Webwatcher: A tour guide for the world wide web. In *PROCEEDINGS OF IJCAI97*, 1997.

[71] M. Jovanovic, F. Annexstein, and K. Berman. Scalability issues in large peer-to-peer networks - a case study of gnutella. Technical report, University of Cincinnati, 2001.

[72] Jason Kincaid. Stylehop's hunt for hot fashion comes to an end as it heads to the deadpool. http://techcrunch.com/2010/12/27/stylehops-hunt-for-hot-fashion-comes-to-an-end-as-it-heads-to-the-deadpool/, December 2010.

[73] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. OceanStore: an architecture for global-scale persistent storage. *ACM SIGPLAN Notices*, 35(11):190–201, November 2000.

[74] D. Kugler. An analysis of GNUnet and the implications for anonymous, censorship-resistant networks. *Privacy Enhancing Technologies. Third International Workshop, PET 2003. Revised Papers. (Lecture Notes in Comput. Sci. Vol.2760)*, pages 161–76, 2003.

[75] S.H. Kwok and K.Y. Chan. An enhanced Gnutella P2P protocol: a search perspective. *Journal of Interconnection Networks*, 5(3):267–78, 2004. Gnutella P2P protocol;peer-to-peer protocol;file sharing;n-phase search;passive search;.

[76] Craig Labovitz. File sharing in the post megaupload era. http://blog.deepfield.net/2012/02/07/file-sharing-in-the-post-megaupload-era/, February 2012.

[77] S. Lam, D. Frankowski, and J. Riedl. Do you trust your recommendations? An exploration of security and privacy issues in recommender systems. *Emerging Trends in Information and Communication Security*, pages 14–29, 2006.

[78] Lawrence Latif. Uk signs acta along with the rest of europe. http://www.theinquirer.net/inquirer/news/2141661/uk-signs-acta-rest-europe, January 2012.

[79] Seunggwan Lee, Daeho Lee, and Sungwon Lee. Personalized dtv program recommendation system under a cloud computing environment. *Consumer Electronics, IEEE Transactions on*, 56(2):1034–1042, May 2010.

[80] Mei Li, Wang-Chien Lien, and A. Sivasubramaniam. Semantic small world: an overlay network for peer-to-peer search. In *12th IEEE Internation Conference on Network Protocols*, pages 228–238, October 2004.

[81] Mei Li, Wang-Chien Lien, and A. Sivasubramaniam. Semantic small world: an overlay network for peer-to-peer search. In *12th IEEE Internation Conference on Network Protocols*, pages 228–238, October 2004.

[82] Kun Liu and Jessica Ryan. Random Projection-Based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining. *IEEE Trans. on Knowl. and Data Eng.*, 18(1):92–106, 2006. Senior Member-Kargupta,, Hillol.

[83] Lu Liu, Nick Antonopoulos, and Stephen Mackin. Managing peer-to-peer networks with human tactics in social interactions. *J. Supercomput.*, 44(3):217–236, 2008.

[84] Ashwin Machanavajjhala, Aleksandra Korolova, and Atish Das Sarma. On the (im)possibility of preserving utility and privacy in personalized social recommendations. Technical Report arXiv:1004.5600, May 2010.

[85] Christopher D. Manning, Prabhakar Raghadan, and Hinrich Schütze. *An Introduction to Information Retrieval.* Cambridge University Press, April 2009.

[86] Ming Mao. Ontology mapping: An information retrieval and interactive activation network based approach ontology mapping: An information retrieval and interactive activation network based approach. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 931–935, October 2008.

[87] Benjamin Markines, Ciro Cattuto, Filippo Menczer, Dominic Benz, and Andreas. Evaluating Similarity Measures for Emergent Semantics of Social Tagging. In *18th International Conference on World Wide Web*, pages 641–650, 2009.

[88] P. Maymounkov and D. Mazieres. Kademlia: a Peer-to-Peer Information System Based on the XOR Metrid. In *IPTPS*, pages 53–65, 2002.

[89] D.A. Menasce. Scalable P2P search. *IEEE Internet Computing*, 7(2):83 – 87, March 2003.

[90] David Meyer. Acta rejected by europe, leaving copyright treaty near dead. http://www.zdnet.com/acta-rejected-by-europe-leaving-copyright-treaty-near-dead-7000000255/, July 2012.

[91] Richard Mortier, Anil Madhavapeddy, Theodore W. Hong, Derek Murray, and Malte Schwartzkopf. Using dust clouds to enhance anonymous communication. 2010.

[92] Rafail Ostrovsky and III. William E. Skeith. A survey of single-database private information retrieval: techniques and applications. In *Proceedings of the 10th international conference on Practice and theory in public-key cryptography*, PKC'07, pages 393–411, Berlin, Heidelberg, 2007. Springer-Verlag.

[93] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT*, 1999.

[94] Pandora. Pandora Radio. www.pandora.com, (visited July, 2012).

[95] Eli Pariser. *The Filter Bubble: What the Internet Is Hiding from You.* Penguin Press, 2011.

[96] Michael Pazzani and Daniel Billsus. Content-based recommendation systems. *The adaptive web*, pages 325–341, 2007.

[97] Michael Pazzani, Daniel Billsus, S. Michalski, and Janusz Wnek. Learning and revising user profiles: The identification of interesting web sites. In *Machine Learning*, pages 313–331, 1997.

[98] Siani Pearson, Yun Shen, and Miranda Mowbray. A privacy manager for cloud computing. In *CloudComp 2009*, number 5931 in Lecture Notes in Computer Science, pages 90–106, 2009.

[99] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips. Tribler: a social-based peer-to-peer system: Research articles. *Concurr. Comput. : Pract. Exper.*, 20(2):127–138, 2008.

[100] J.A. Pouwelse, J. Yang, M. Meulpolder, D.H.J. Epema, and H.J. Sips. Buddycast: An operational peer-to-peer epidemoc protocol stack. In *Fourteenth Annual Conference of the Advanced School for Computing and Imaging*, June 2008.

[101] Michael O. Rabin. Efficient dispersal of information for security, load balancing and fault tolerance. *Journal of the ACM*, 36(2):335 – 348, April 1989.

[102] Kadangode K. Ramakrishan, Divesh Srivastava, Tae Won Cho, and Yin Zhang. User-Powered Recommendation System, June 2010.

[103] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: output perturbation for queries with joins. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 107–116. ACM, 2009.

[104] David Rebollo and Jordi Forné. Optimized Query Forgery for Private Information Retrieval. volume 56 of *IEEE Trans. Inform. Theory*, pages 4631–4642, September 2010.

[105] M.K. Reiter and A.D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[106] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *1994 ACM Conference on Computer Supported Collaborative Work Conference*, pages 175–186, Chapel Hill, NC, 10/1994 1994. Association of Computing Machinery, Association of Computing Machinery.

[107] Retroshare Team. Retroshare: secure communications with friends. http://retroshare.sourceforge.net/, (visited July, 2012).

[108] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[109] M. Rogers and S. Bhatti. How to disappear completely: A survey of private peer-to-peer networks. *RN*, 7(13):1, 2007.

[110] Giancarlo Ruffo and Rossano Schifanella. Evaluating peer-to-peer recommender systems that exploit spintaneous affinities. In *SAC*, 2007.

[111] Magnus Sahlgren and Rickard Cöster. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 487–493, August 2004.

[112] J. Ben Schafer, Joseph Konstan, and John Riedi. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, EC '99, pages 158–166, New York, NY, USA, 1999. ACM.

[113] Rossano Schifanella, André Panisson, Cristina Gena, and Giancarlo Ruffo. Mobhinter: epidemic collaborative filtering and self-organization in mobile ad-hoc networks. In *ACM Conference on Recommender systems*, pages 27–34, New York, NY, USA, 2008. ACM.

[114] Berry Schoenmakers and Pim Tuyls. Practical Two-Party Computation Based on the Conditional Gate. In *Advances in Cryptology - ASIACRYPT 2004*, volume 3329/2004 of *Lecture Notes in Computer Science*, pages 119–136. Springer Berlin / Heidelberg, 2004.

[115] Shopping Guide GmbH. Ciao! Prize comparision and consumer reviews. http://www.ciao.co.uk/, (visited July, 2012).

[116] Chris Sosa, Blake C. Sutton, and H. Howie Huang. PicFS: The Privacy-enhancing Image-based Collaborative File System. 2010.

[117] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. Efficient content location using interest-based locality in peer-to-peer systems. *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies.*, 3(3):2166– 2176, March 2003.

[118] Stockholm District Court. Verdict B 13301-06, 17 April 2009, April 2009.

[119] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.

[120] Inc. Strands. Strands Recommender. http://recommender.strands.com/, (visited March, 2012).

[121] StumbleUpon. StumbleUpon. http://www.stumbleupon.com, (visited March, 2012).

[122] X. Su and T.M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4, 2009.

[123] K. Tatara, Y. Hori, and K. Sakurai. Query forwarding algorithm supporting initiator anonymity in GNUnet. *Proceedings. 11th International Conference on Parallel and Distributed Systems*, Vol. 2:235–9, 2005.

[124] Luis Teran and Andreas Meier. A Fuzzy Recommender System for eElections. In *Electronic Government and the Information Systems Perspective*, volume 6267 of *Springer-Verlag*, pages 67–76, 2010.

[125] Rajesh Thiagarajan, Geetha Manjunath, and Markus Stumptner. Finding experts by semantic matching of user profiles. In *3rd Expert Finder Workshop on Personal Identification and Collaborations: Knowledge Mediation and Extraction*, October 2008.

[126] Alvin Toffler. *The Third Wave.* Bantam, 1984.

[127] Trade European Commission. The Anti-Counterfeiting Trade Agreement (ACTA). http://ec.europa.eu/trade/creating-opportunities/trade-topics/intellectual-property/anti-counterfeiting/, April 2010.

[128] Bao Ngoc Tran and Thuc Dinh Nguyen. A Graph Isomorphism Bases Authentication Protocol for Access Control in WLAN. In *22nd International Conference on Advanced Information Networking and Applications*, 2008.

[129] Duc Thanh Tran, Stephan Bloehdorn, Philipp Cimiano, and Peter Haase. Expressive resource descriptions for ontology-based information retrieval. In *Proceedings of the 1st International Conference on the Theory of Information Retrieval (ICTIR'07), 18th - 20th October 2007, Budapest, Hungary*, pages 55–68, 2007.

[130] Trust Pilot. Trust Pilot. Your review informs other shoppers. http://www.trustpilot.com/, (visited July, 2012).

[131] U.S. Department of Justice. Justice Department Charges Leaders of Megaupload with Widespread Online Copyright Infringement. http://www.fbi.gov/news/pressrel/press-releases/justice-department-charges-leaders-of-megaupload-with-widespread-online-copyright-infringement, January 2012.

[132] USLegal. Plausible Deniability Law & Legal Definition. http://definitions.uslegal.com/p/plausable-deniability/, 2011.

[133] Juan Vera-del Campo, Ana González-Muro, and Miguel Soriano. Private Audio Streaming for an Automated Phone Assistance System. In *Sixth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, October 2011.

[134] Juan Vera-del Campo, Juan Hernández-Serrano, and Josep Pegueroles. Análisis de Seguridad en un Sistema de Archivos Distribuido. In *X Reunión Española sobre Criptografía y Seguridad de la Información (RECSI)*, 2008.

[135] Juan Vera-del-Campo, Juan Hernández-Serrano, and Josep Pegueroles. SCFS: Design and Implementation of a Secure Distributed Filesystem. In *SECRYPT*, 2008.

[136] Juan Vera-del Campo, Juan Hernández-Serrano, and Josep Pegueroles. SCFS: lewis.upc.es/svn/dfs, 2008.

[137] Juan Vera-del Campo, Juan Hernández-Serrano, and Josep Pegueroles. Profile-based searches on P2P social networks. In *The Ninth International Conference on Networks, ICN*, 2010.

[138] Juan Vera-del-Campo, Josep Pegueroles, Juan Hernández-Serrano, and Miguel Soriano. Design of a p2p content recommendation system using affinity networks. *Computer Communications*, Accepted, 2012.

[139] Juan Vera-del Campo, Josep Pegueroles, and Miguel Soriano. Demonstration of Security in Service Discovery and Access for Ubiquitous Networks. In *Demonstration session of the 25th conference on computer communications IEEE INFOCOM*, 2006.

[140] Juan Vera-del Campo, Josep Pegueroles, and Miguel Soriano. Providing Security Services in a Multiprotocol Service Discovery System for Ubiquitous Networks. In *First Conference on Availability, Reliability and Security (ARES)*, 2006.

[141] Juan Vera-del Campo, Josep Pegueroles, and Miguel Soriano. Seguridad en Protocolos de Descubrimiento de Servicio. In *IX Reunión española sobre Criptografía y Seguridad de la Información (RECSI)*, 2006.

[142] Juan Vera-del Campo, Josep Pegueroles, and Miguel Soriano. MSD: A Middleware for Secure Service Discovery in Pervasive and Mobile Computing Environments. *Journal of Networks*, 2007.

[143] E. Vozalis and K. G. Margaritis. Analysis of Recommender Systems' Algorithms. In *Sixth Hellenic-European Conference on Computer Mathematics and its Applications (HERCMA)*, pages 732–745, 2003.

[144] J. Wang, A.P. De Vries, and M.J.T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

[145] Duncan J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton Studies on Complexity, 2003.

[146] R. R. Yager. Fuzzy Logic methods in recommender systems. In *Fuzzy Sets and Systems*, pages 133–149, 2003.

[147] Wai Gen Yee, Dongmei Jia, and Ophir Frieder. Finding rare data objects in p2p file-sharing systems. *Proceedings - Fifth IEEE International Conference on Peer-to-Peer Computing, P2P 2005*, 2005:181–190, 2005.

[148] Yelp Inc. Yelp. Real people, real reviews. http://www.yelp.com/, (visited July, 2012).

[149] J. Zhan, C.L. Hsieh, I.C. Wang, T.S. Hsu, C.J. Liau, and D.W. Wang. Privacy-preserving collaborative recommender systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(4):472–476, 2010.

[150] F. ZHANG, J. WANG, and J. CHAO. Cross-system Privacy Preserving Recommendation Algorithm Based on Secure Multi-party Computation. *Journal of Computational Information Systems*, 6(9):3013–3021, 2010.

[151] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 506–515. IEEE, 2008.

[152] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. Solving the aparent diversity-accuracy dilemma of recommender systems. In *Proceedings of the National Academy of Sciences of the USA*, volume 107, pages 4511–4515, 2010.