CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS

David Pàmies Estrems

UNIVERSITAT
ROVIRA i VIRGILI

Contributions to Lifelogging Protection In Streaming Environments

---

DAVID PÀMIES-ESTREMS

DOCTORAL THESIS
2020

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

David Pàmies-Estrems

CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS

DOCTORAL THESIS

supervised by Dr. Jordi Castellà-Roca
and Dr. Joaquin Garcia-Alfaro

Departament d'Enginyeria Informàtica i Matemàtiques

UNIVERSITAT ROVIRA i VIRGILI

Tarragona, Catalonia, Spain
May 2020

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

UNIVERSITAT ROVIRA i VIRGILI

FEM CONSTAR que aquest treball, titulat "Contributions to Lifelogging Protection In Streaming Environments", que presenta en David Pàmies-Estrems per a l'obtenció del títol de Doctor, ha estat realitzat sota la nostra direcció al Departament d'Enginyeria Informàtica i Matemàtiques d'aquesta universitat.

HACEMOS CONSTAR que el presente trabajo, titulado "Contributions to Lifelogging Protection In Streaming Environments", que presenta David Pàmies-Estrems para la obtención del título de Doctor, ha sido realizado bajo nuestra dirección en el Departamento de Ingeniería Informática y Matemáticas de esta universidad.

WE STATE that the present study, entitled "Contributions to Lifelogging Protection In Streaming Environments", presented by David Pàmies-Estrems for the award of the degree of Doctor, has been carried out under our supervision at the Department of Computer Engineering and Mathematics of this university.

Tarragona, May, 2020

Doctoral Thesis Supervisor

Jordi Castellà-Roca

Doctoral Thesis Supervisor

Joaquin Garcia-Alfaro

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

"What are the roots that clutch, what branches grow
Out of this stony rubbish? Son of man,
You cannot say, or guess, for you know only
A heap of broken images, where the sun beats,
And the dead tree gives no shelter, the cricket no relief,
And the dry stone no sound of water. Only
There is shadow under this red rock,
(Come in under the shadow of this red rock),
And I will show you something different from either
Your shadow at morning striding behind you
Or your shadow at evening rising to meet you;
I will show you fear in a handful of dust."

T.S. Eliot

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

Acknowledgements

To begin, I would especially like to thank the work done by my two advisors, Jordi Castellà-Roca and Joaquin Garcia-Alfaro. As can be deduced from their career, they are top-level experts in this field of knowledge. Without them, this experience would not have been so interesting.

I can not help but mention the members of CRISES research group of the Universitat Rovira i Virgili and his head Josep Domingo-Ferrer. The contributions of Alexandre Viejo and David Sánchez. The communtication skills and the goodwill of Carles Anglés. The diligence of Jesús Manjón and the support of Jordi Ribes.

In other URV departments I have also found people who have helped me during those years. In particular I would like to thank the time and efforts that Aïda Valls has dedicated to me on several occasions, as well as the advice of Jordi Duch, Pedro Garcia, Carles Barberà and Agustí Solanas along this path.

Je voudrais également exprimer ma gratitude à Télécom SudParis, partie de l'Institut Polytechnique de Paris, en particulier au Département Réseaux et Services de Télécom et à son responsable, Hervé Debar, qui m'ont très gentiment accueilli pour des mois passionnants. Je voudrais souligner en particulier le soutien de Maryline Laurent et Nesrine Kaaniche pour leur collaboration dans la recherche. Antoine Bernard et Houda Ibn-Khedher pour leur hospitalité infinie. Les conseils précieux et les discussions fructueuses avec Grégory Blanc, Jose Rubio, Anna Guinet, Ender Alvarez, Fabien Charmet, Mohammed El Barbori et Mustafizur Shahid. J'aimerais également remercier Sandra Gchweinder pour son aide dans la gestion. Et surtout pour les bons moments partagés avec Mariana Segovia, Thomas Tournaire et Vu Thai Duong. C'était un privilège de faire partie d'une des meilleures écoles d'ingénieurs françaises. Merci beaucoup !

I must also acknowledge the various organizations that partially supported this work, including the Department of Computer Science and Mathematics of the Universitat Rovira i Virgili, the CRISES Research Group, the UNESCO Chair in Data Privacy and CYBERCAT-Center for Cybersecurity Research of Catalonia.

Last but not least, I must express my gratitude to my friends and family. Joan, Andrew, Raida, Tere, Ramon, Maite, Arlet and Laura. Your support and ability to collect blackberries were especially useful to survive along the way.

UNIVERSITAT ROVIRA I VIRGILI

Abstract

Escola Tècnica Superior d'Enginyeria
Departament d'Enginyeria Informàtica i Matemàtiques

Doctor of Philosophy

by David Pàmies-Estrems

Every day, more than five billion people generate some kind of data over the Internet. As a tool for accessing that information, we need to use search services, either in the form of Web Search Engines or through Personal Assistants. On each interaction with them, our record of actions via logs, is used to offer a more useful experience. For companies, logs are also very valuable since they offer a way to monetize the service.

Monetization is achieved by selling data to third parties, however query logs could potentially expose sensitive user information: identifiers, sensitive data from users (such as diseases, sexual tendencies, religious beliefs) or be used for what is called "life-logging": a continuous record of one's daily activities. Current regulations oblige companies to protect this personal information. Protection systems for closed data sets have previously been proposed, most of them working with atomic files or structured data. Unfortunately, those systems do not fit when used in the growing real-time unstructured data environment posed by Internet services.

This thesis aims to design techniques to protect the user's sensitive information in a non-structured real-time streaming environment, guaranteeing a trade-off between data utility and protection. In this regard, three proposals have been made in efficient log protection. The first is a new method to anonymize query logs, based on probabilistic k-anonymity and some de-anonymization tools to determine possible data leaks.

A second method has been improved in terms of a configurable trade-off between privacy and usability, achieving a great improvement in terms of data utility.

Our final contribution concerns Internet-based Personal Assistants. The information generated by these devices is likely to be considered life-logging, and it can increase the user's privacy risks. The proposal is a protection scheme that combines log anonymization and sanitizable signatures.

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

Contents

List of Figures

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

List of Tables

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

Abbreviations

AOL	**A**merica **O**n**L**ine
CPU	**C**entral **P**rocessing **U**nit
EMD	**E**arth **M**over's **D**istance
GDPR	EU **G**eneral **D**ata **P**rotection **R**egulation
HIPAA	**H**ealth **I**nsurance **P**ortability and **A**ccountability **A**ct
IR	**I**nformation **R**etrieval
ICT	**I**nformation and **C**ommunication **T**echnology
IT	**I**nformation **T**echnologies
K-S	**K**olmogorov-**S**mirnov
LBQ	**L**ocation **B**ased **Q**uery
MDAV	**M**aximum **D**istance **A**verage **V**ector
MS	**M**ain **S**ervice
NLP	**N**atural **L**anguage **P**rocessing
NLTK	**N**atural **L**anguage **T**ool**K**it
ODP	**O**pen **D**irectory **P**roject
P3P	**P**latform for **P**rivacy **P**references
PA	**P**ersonal **A**ssistant
PET	**P**rivacy-**E**nhancing **T**echnologies
PIR	**P**rivate **I**nformation **R**etrieval
PPT	**P**robabilistic **P**olynomial-**T**ime
SD	**S**tandard **D**eviation
SDC	**S**tatistical **D**isclosure **C**ontrol
SQL	**S**tructured **Q**uery **L**anguage
SS	**S**ocial **S**ecurity

SSD	**S**olid **S**tate **D**isk
SU	**S**emantic **U**nit
TP	**T**hird **P**arty
URL	**U**niform **R**esource **L**ocator
WSE	**W**eb **S**earch **E**ngine
WWW	**W**orld **W**ide **W**eb

Outline of the Thesis

This thesis is organized in 6 chapters.

Chapter 1 states the thesis' aims and objectives. It also outlines the direction taken in our research, and introduces the results contained in the remainder of the thesis.

Chapter 2 summarizes the most relevant and up to date research conducted in this field. It briefly explains the concepts of privacy and anonymization, studying in detail the main techniques applied in fixed-length and data-stream input treatment.

Chapter 3 proposes a new method to anonymize query logs based on k-anonymity. It also proposes some de-anonymization tools to determine possible privacy problems, in case an attacker gains access to the anonymized query logs. This approach tries to preserve the original user interests, but spreads possible semi-identifier information over many users, to prevent linkage attacks.

Chapter 4 presents an alternative method to anonymize query logs, with the goal of obtaining the same or better level of privacy, in addition to a customizable data utility through the use of parameterizable levels of categorization. It also devises formal and experimental proofs that ensure its feasibility in terms of privacy, utility, and scalability achievement.

Chapter 5 highlights the threats generated by the use of Internet-based personal assistants. Although the interaction with those devices is conducted by voice in a local environment, the generated logs are often collected and stored on remote servers, which can lead to serious privacy risks. To mitigate privacy threats, this chapter proposes a protection scheme that combines log anonymization and sanitizable signatures.

Chapter 6 presents the main conclusions of the thesis, including achieved goals, limitations of the current approach and avenues for future research.

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

A la meva mare.

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

Chapter 1

Introduction

This chapter is divided into three main sections. In section 1.1 we briefly introduce the conjunction of the facts that motivates the direction taken in our research. In section 1.2 we give a general description of each of the contributions of this thesis. Finally, section 1.3 lists the publications emanating from this thesis.

1.1 Motivation

The whole data created, captured, or replicated, is called the Global Datasphere. IDC predicts that the Global Datasphere will grow at an annual rate of 61%, from 33 Zettabytes (ZB) in 2018 to 175 ZB by 2025. In terms of population, more than five billion people generate some kind of data every day. By 2025, that number will be six billion, or 75% of the world's population, and each connected person will have at least 5 000 daily digital interactions, in front of the current 800 [1].

Global Datasphere is composed of very varied data sources, such as surveillance cameras, medical data, digital TV/radio, video games, mobile phones, VoIP, PCs, servers' logs, and embedded systems (in automobiles, vending machines, etc.).

If we focus only on the Web, individuals usually interact with what is known as a Web Search Engine (WSE), a system that is designed to search the World Wide

Web in a systematic way for particular information specified in a textual search query [2]. These queries contain certain keywords related to the topics they are looking for. The WSE, then retrieves and presents an accurate list of responses that tackle those topics. Generally, people use these services mainly for one of these three reasons: information research, shopping, or entertainment.

Popularity of these tools remain very high [3] since only sending a basic query allows people to find, with a minimal response time, what they are looking for. Therefore, it can be assumed that services such as the ones offered by Google or Bing will continue to be essential in the coming years.

Apart from the information retrieval process needed to fulfill the user's query, the service also starts a data gathering process that stores the submitted query (i.e., the keywords) and some metadata e.g. date of the query, some identifiers of the query issuer (user id), which specific search result was selected by the issuer (if applies), and other user related information [4, 5]. Additionally, queries made on most modern devices often send the user location and the local time as two additional parameters when it comes to find the most convenient information in each situation. Therefore, user, location, and local time are also registered in the company servers. This information is usually registered in the form of a log, and all the logs are usually stored in a file or database, which contains all the previously recorded logs, and is generally referred to as *query logs* [6]. Over time, the *query logs* gathering process, produces a huge data repository of information about our everyday lives, which can be called *lifelogging* [7].

In the past, people have only used Web Search Engines as the main gateway to the Internet, but as time goes by, we can find more alternatives. The new proposals are trying to reduce the barriers to access information even more and to make it accessible to everyone, anytime. As a consequence of these innovations, today we can find a multitude of technological tools that have been developed precisely for this reason: smartphones, smartgateways, smartwatches and activity bands. As a whole, we will refer to these devices as Personal Assistants. For reasons of economies of scale, the development of this type of device is only available to a

few technological organizations [8, 9]. These tools allow the *lifelogging* process to be even more exhaustive.

Furthermore, the data about individuals, groups, and periods of time could be combined into bigger groups over longer periods of time; what has been recently designated as *Big Data*.

Once the query logs are generated, they are processed and analyzed to build users' profiles [10] which could be used to improve their services in several ways. In the literature, a user profile is generally considered a set of well-defined categories of interests (e.g. science, health, society, sports, etc.) with a certain weight assigned to each one according to the evidence generated by the corresponding user and how they have been classified under each category [11]. When focusing on search services, search queries are the evidence, and the amount of queries from each user classified under each category reflects the related weight.

Users' profiles could be helpful in improving the service offered in the following ways:

- **Personalization**. Query terms can have multiple meanings; therefore, identifying the sense required by the user represents a challenge. In order to better fulfill their needs, more relevant results should be placed in the first position of the returned results. By analyzing the queries submitted by users in the past, the service providers can gain knowledge about their preferences, and this knowledge can be used to contextualize and disambiguate terms in the future [12, 13]. This way, if a user searches for "Mercury" and her profile indicates that she is interested in "Astronomy", the WSE can prioritize relevant results (e.g., URLs) that correspond to the planet Mercury in the initial positions of search results, instead of the chemical element.

- **Usability**: By knowing the frequencies of most formulated queries and most selected results, online services are able to improve the ranking algorithms [14]. This can also be used to suggest alternative queries [15]. Such suggestions can show how to correct mistakes when typing, add specificity

to the initial query, or provide similar queries with more results after retrieving poor results for the original query submission [16]. Continuing with the example of the term "Mercury", if a user inputs just this term, the service could suggest more specific alternative queries, such as "Mercury - Planet", "Mercury - Element", "mercury poisoning", and "mercury marine", among others. These alternative keywords are expected to retrieve more accurate results for the user.

However, query logs can also be exploited for other purposes because they reveal powerful insights about customer intent-to-purchase and other factors [17]. This new exploitation can be conducted by the service provider itself or by a third party, for the following purposes:

- **Marketing**: While characterizing general user profiles, user behavior, and user search habits, it is possible to improve keyword advertising campaigns and extract market tendencies among others. For example, the user can be characterized by their query logs (gender, age, income, education, etc.) and afterwards verify if the advertisements have had an impact on the intended audience (interests and behaviour) [18, 19]. Besides this, it is possible to extract market tendencies [20]. More concretely, search analytics is one of the cornerstones of so-called *Search Engine Marketing* and it is in charge of using search data to investigate particular interactions among searchers, the search engine, or content during searching episodes [21]. For example, Google Trends[1] is a service that exploits this kind of data. In particular, Google Trends shows how often a particular term is searched according to the total search volume across various regions of the world.

- **Research**: As stated in [22], query logs contain information that would never be available to researchers using conventional data collection techniques. For example, a medical researcher might discover that people with

[1]Google Trends. `https://www.google.com/trends/`

asthma tend to wear wool or live in areas with coal power plants; a sociologist may study how ideas spread from one person to an entire community, or may investigate the differences between the interests that individuals claim to have during face-to-face interviews and the real interests that their search queries reveal [23]; a political scientist might learn about democracy by studying the evolution of political searches by users in a developing country. Query logs also enable researchers to ask questions that would normally require going backward in time. For example, a medical researcher might study people diagnosed with diabetes today to find out what their primary symptoms were six months ago. Asking them directly, once they learn they have diabetes, may result in subjective bias [22]. Last but not least, a computer scientist may study and analyze new Information Retrieval (IR) algorithms via a common benchmark query log [24]. It may also be used to learn about users information needs and query formulation approaches [20] and revolve around the use of language in queries [25], among other research topics [26–30].

Turning this data into knowledge is a key interest for many companies, since the extraction of valuable information is what enables those companies to obtain economic benefits. This process is known as *data monetization* [31]. A classic example of data monetization can be found in the supermarket group Tesco [32], which uses a data ecosystem to monetize data from its loyalty program, the Tesco ClubCard. In return for opting in to ClubCard, customers receive personalized offers. The same knowledge could be extracted from the queries against Internet services, making that information valuable. In addition to those benefits, building and exploiting query logs may lead to serious privacy problems as well, since this process enables these organizations to compile a lot of data about individuals. Each query logged contains a user identifier, a text about what the user is looking for, the time when it was done, the results selected by the user, and in some cases, the user's location.

The keywords of each query and the related metadata may provide sensitive information from the user such as behaviors, habits, interests, religious views, sexual orientation, etc. to anyone who has access to the logs and therefore to third parties. Even more, some query keywords may contain identifiers and quasi-identifiers [33], which may allow to link queries with real people. This is specially feasible, given current tendencies such as *vanity search* and *egosurfing* [34], in which people look for their own names over the Internet.

Someone might think that if it is too dangerous to save these logs, a possible solution would be to not store any of them. However, as we have seen, the service they offer is essential in order to allow people to find information on the Internet, and the economic viability of the companies that offer this service is closely linked to the data monetization process. Therefore, they will continue keeping the data.

The current practice was recently put in the spotlight by the U.S. Federal Trade Commission when it published a report about data collection and use practices of the most relevant Data Brokers [35]. The widespread development of technologies such as *Data Mining* and *Artificial Intelligence* over *Big Data*, make the task of extracting information, knowledge, or relevant relationships easier every day [36]. This can lead to very serious privacy risks of personal data disclosure, as this data can be exploded not only in isolation, but also as a combination of information generated between several sources.

Anyone who uses these services is constantly disclosing, in a direct or indirect way, personal data to the organization which provides the service. Using Personal Assistants, the situation is even more accentuated, since the user is not in front of a computer. Users tend to establish more relaxed relationships with the device, sometimes without even knowing if it is working or sending information to another site, and mostly seeing it as a friend or an extension of its person.

Therefore there is a strong need to protect sensible users' data, while allowing the resulting data to remain useful for companies that offer the service.

To protect the privacy of individuals, several measures were established by lawmakers, from the USA Privacy Act of 1974 [37], until the recent General Data Protection Regulation (GDPR), adopted by the European Union in 2016 [38]. Specifically in this last case, all entities that perform personal data processing related to European citizens, must be able to demonstrate at any time that they are complying with the regulations established by the GDPR.

Although query logs could be protected prior to their publication, there is no absolute guarantee of anonymity. To protect the user's identity and sensitive data, there exist some techniques that are able to eliminate direct user identifiers. However, a specialized type of attack, called Record Linkage attack, allows the linking of different user records, which contain seemingly harmless information, but when all the data can be related to, it can end up revealing sensitive information from the users [19, 39]. As an example, there is one well-known case, the AOL Research scandal, in which around 36 million queries performed by AOL's customers were publicly disclosed. Although records were previously de-identified, it was possible to identify some users from the disclosed query logs and other sensitive information was exposed [40]. The case ended up with an important damage to AOL users' privacy and to AOL itself, with several class action law suits and complaints against the company [41–43].

In addition, these data protection schemes have traditionally been constructed from a static point of view; that is, a closed data set taken and protected in isolation. These techniques may be improved to reach a good level of protection, but our context seems different. Interactions made by a user and therefore their history, increase frequently. This could be seen in global terms as a real-time data stream.

As indicated by [1], real-time data currently represents 15% of the Global Datasphere, but predictions point that nearly 30% will be real-time by 2025. Enterprises looking to provide superior customer experience and grow share must have data infrastructures that can meet this growth in real-time data. Therefore, what we need to propose is a streaming protection scheme.

More specifically, we propose using a server-side software capable of processing queries in real time and building anonymized query logs that still retain enough data utility to allow its monetization. As a result, search service providers may then offer the protected query logs to external organizations for data monetization purposes while keeping the real query logs in a safe place; otherwise, search service providers may also decide to only keep protected logs, getting in turn a lower risk of information disclosure in case of a direct attack.

At the same time, search service providers may generate profiles that contain similar queries from other users. Each profile is not the user's profile, but it contains the same interests, with the benefit that quasi-identifiers do not correspond with this user.

We also want to study an additional case, that may remain vulnerable: lifelogging data streams generated by Internet-based personal devices like Google Home and Amazon Echo [5, 44]. The issue generated by such devices, related to other data information actors in terms of EU data protection directives, is an interesting topic which may need some customised solution, therefore we will scrutinize that case in detail.

To sum up, this thesis proposal takes into account the role of the organizations and their needs to monetize the user's data, and aims to generate streams of query logs in real time that preserve an adequate level of data utility for the monetization process, but at the same time with a limited privacy disclosure risk.

1.2 Research Contributions

In this section we briefly describe each of the contributions of this thesis. In the order of appearance in the remainder of the text, the research contributions are: Working at the Web Search Engine Side to Generate Privacy-preserving User Profiles, A Real-Time Query Log Protection Method for Web Search Engines, Lifelogging Protection Scheme for Internet-Based Personal Assistants.

1.2.1 Working at the Web Search Engine Side to Generate Privacy-preserving User Profiles

The popularity of Web Search Engines (WSEs) enables them to generate a lot of data in form of query logs. These files contain all search queries submitted by users. Economical benefits could be earned by means of selling or releasing those logs to third parties. Nevertheless, this data potentially expose sensitive user information. Removing direct identifiers is not sufficient to preserve the privacy of the users. Some existing privacy-preserving approaches use log batch processing but, as logs are generated and consumed in a real-time environment, a continuous anonymization process would be more convenient. In this way, in this paper we propose: i) a new method to anonymize query logs, based on k-anonymity; and ii) some de-anonymization tools to determine possible privacy problems, in case that an attacker gains access to the anonymized query logs. This approach preserves the original user interests, but spreads possible semi-identifier information over many users, preventing linkage attacks. To assess its performance, all the proposed algorithms are implemented and an extensive set of experiments are conducted using real data.

1.2.2 A Real-Time Query Log Protection Method for Web Search Engines

Web search engines (e.g., Google, Bing, Qwant, and DuckDuckGo) may process a myriad of search queries per second. According to Internet Live Stats, Google handles more than two hundred million queries per hour, i.e., about two trillion queries per year. For monetization purposes, the queries can be stored and complemented with additional data, referred to as query logs. Together, they can correlate valuable information to build accurate user profiles. Before releasing the query logs to third parties (e.g., for profit purposes), the personal information contained in the query logs must be properly protected by the web search engines. Current regulations establish strict control, and require from provable anonymization processing

(e.g., in terms of statistical disclosure) of any personally identifiable information. In this paper, we tackle this challenge. We propose a real-time anonymization solution to protect streams of unstructured data at the server side. Our approach is based on the use of a probabilistic k-anonymity technique. It allows probabilistic processing of personally identifiable attributes contained in the query logs, with provable privacy properties. Our solution handles limitations of traditional k-anonymity approaches with respect to unstructured data and real-time processing. We present the implementation of our solution and report experimental evaluation results. The evaluation is conducted in terms of privacy, utility, and scalability achievement. Results validate the feasibility of our proposal.

1.2.3 Lifelogging Protection Scheme for Internet-Based Personal Assistants

Internet-based personal assistants are promising devices combining voice control and search technologies to pull out relevant information to domestic users. They are expected to assist in a smart way to household activities, such as to schedule meetings, find locations, reporting of cultural events, sending of messages and a lot more. The information collected by these devices, including personalized lifelogs about their corresponding users, is likely to be stored by well-established Internet players related to web search engines and social media. This can lead to serious privacy risks. The issue of protecting the identity of domestic users and their sensitive data must be tackled at design time, to promptly mitigate privacy threats. Towards this end, this paper proposes a protection scheme that jointly handles the aforementioned issues by combining log anonymization and sanitizable signatures.

1.3 Publication List

The following publications and preprints form the core of this thesis.

Publications:

1. David Pàmies-Estrems, Jordi Castellà-Roca, and Alexandre Viejo. Working at the Web Search Engine Side to Generate Privacy-preserving User Profiles. *Expert Systems with Applications*, 64(C):523–535, December 2016. ISSN 0957-4174. doi: 10.1016/j.eswa.2016.08.033. URL https://doi.org/10.1016/j.eswa.2016.08.033

2. David Pàmies-Estrems, Nesrine Kaaniche, Maryline Laurent, Jordi Castella-Roca, and Joaquin Garcia-Alfaro. Lifelogging Protection Scheme for Internet-based Personal Assistants. In Joaquin Garcia-Alfaro, Jordi Herrera-Joancomartí, Giovanni Livraga, and Ruben Rios, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology (CBT 2018 and DPM 2018)*, pages 431–440, Cham, September 2018. Springer. ISBN 978-3-030-00305-0. doi: 10.1007/978-3-030-00305-0_31. URL https://doi.org/10.1007/978-3-030-00305-0_31

3. David Pàmies-Estrems, Jordi Castellà-Roca, and Joaquin Garcia-Alfaro. A Real-Time Query Log Protection Method for Web Search Engines. *IEEE Access*, May 2020. doi: 10.1109/ACCESS.2020.2992012. URL https://doi.org/10.1109/ACCESS.2020.2992012

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

Chapter 2

State of the Art

Privacy in our society is increasingly critical. Mobile and computing applications are dramatically increasing the amount of personal data released to service providers as well as to third parties [48]. This data often includes the location of individuals, their movement patterns as well as sensor-acquired data that may reveal individuals' physical conditions and habits. Privacy solutions must comply to ethical and legal requirements, and not prevent profitable business models.

The illegitimate use of personal data is becoming a commonplace. We can find varied examples. In the AOL Research case [49], a detailed search logs by AOL were released. The release was intentional and intended for research purposes; however, the public release meant that the entire Internet could see the results rather than a select number of academics. AOL did not identify any user in the report, and in fact the logs were previously protected, but personally identifiable information was present in many of the queries. This allowed to identify some user and sensible information from the query logs [40]. This case underscores how much people unintentionally reveal about themselves when they use search engines, and how risky it can be for companies like AOL, Google and Yahoo to compile such data. AOL itself, has been sued over its online release of data, accused of violating the Electronic Communications Privacy Act and of fraudulent and deceptive business practices, among other claims [42]. The lawsuit seek at least

$5,000 for every person whose search data was exposed. Two AOL employees were fired and the Chief Technology Officer resigned over the incident.

In fact, other companies are also not exempt from controversy. USA Justice Department request data about user's search queries to America Online, Yahoo, Microsoft an Google [43]. That case did not involve information that could be linked to individuals, but highlights what Internet user can expect for the data trail they leave online.

In a similar vein we can find other cases concerning privacy, such as NSA PRISM program revealed by E. Snowden in May 2013[1], or the recent scandal of Cambridge Analytica and Facebook, published in April 2018[2].

Our work is focused on to the use of *privacy-enhancing technologies* (PETs) applied to the web search paradigm. In this chapter, we summarise most relevant research conducted in this field and published up to date. The responsibility of applying PETs relies on two main actors: user and WSE. Notwithstanding, we have chosen to organize the research in three main groups as shown on Figure 2.1, on the basis of previous classifications [3, 50, 51]. All the proposals are designed to protect the user's privacy in front of WSEs.

The first group, studied in detail in Section 2.1, contains proposals that protect user's privacy at the WSE side without the need for user's participation. They are asynchronous and transparent to the user. The second group, described in Section 2.2, includes approaches that protect user's privacy without any help from the WSE, i.e., when user do not require any changes at the server side of the WSE. The third group, in Section 2.3 comprises approaches that require a certain level of cooperation between the user and the WSE. The latter are not considered as server-side, since the user actively participate in the process — when WSEs do not cooperate, it is assumed that the user immediately detect them. In the sequel, we report related work under all three categories.

[1]http://www.bbc.com/news/world-us-canada-23123964
[2]http://fortune.com/2018/04/10/facebook-cambridge-analytica-what-happened/

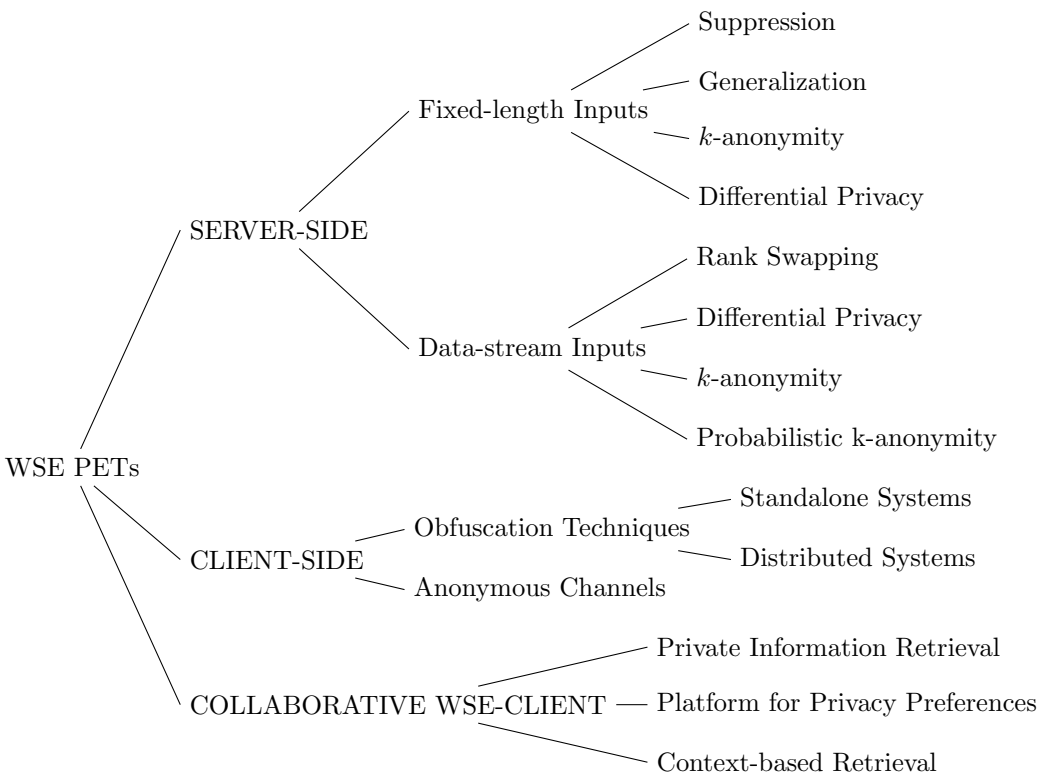

FIGURE 2.1: Classification of Web Search Engine Privacy Enhancing Technologies (PETs)

2.1 Survey on Server Side Proposals

Anonymization of query logs on Server Side has been addressed in the literature from different approaches. WSEs aim at anonymizing data while minimizing information loss, with the goal of commercialize releases of the protected set of query logs to third-parties. Therefore, our work is focused on this assumption. Anonymization solutions to reach such a goal can get classified according to anonymizaiton inputs. Most solutions are either processing fixed-length (e.g., block-based) or data-stream inputs.

2.1.1 Fixed-length input

Chronologically, the fixed-length input methods were the first to be proposed. Some statistic and/or semantic methods are applied to the data-set to find the sensible information and reduce the sensible information disclosure.

Existing proposals of this type, consider a set of finite and static data structures, therefore, each set contains all the elements before beginning the treatment. To make the protection of the whole dataset, the protection is conducted as a two-step process, first all the dataset elements are analyzed and then a comprehensive protection can be applied where needed.

The interesting thing about this approach is that some logs may not divulge information for themselves, but once some logs of the same user are related, the aggregated data may reveal some additional sensible information about the individual. By having all the data available, it allows those methods to process all the queries and strategically modify or remove queries to reduce the risk of information disclosure, before publishing any data.

A possible drawback of this approach can be found by anonymizing twice the same data and obtaining slightly different results. In this case, each additional anonymization would be releasing additional information about the original data.

Another limitation of these approaches is that they need to complete the data set, before being able to treat it and release it. These sets can be large and slow to complete. Therefore, in some scenario where data is needed continuously, this approach is not feasible. Additionally, if periodic blocks of information are freely released, these approaches may also reveal certain temporary information related to the user.

Below are explained the main approaches which belong to the fixed-length input category, divided in four main subcategories, depending of the applied technique: Suppression, Generalization, k-anonymity and Differential Privacy.

2.1.1.1 Suppression

Considering that the data to be treated is in fixed-length input, a possible first approach to achieve query log anonymization is to review all that data and eliminate or hash those elements which, in isolation or combination, may reveal sensible

information. The analysis of the dataset assumes either statistic or semantic methods, to identify which elements require suppression.

Examples of suppression under the context of query logs anonymization exist in the related literature [16]. In particular, they pose seven proposals: Log deletion, hashing queries, identifier deletion, hashing identifiers, scrubbing query content, deleting infrequent queries and shortening sessions. Those include some basic statistic and semantic methods, concluding that search industry, academia, consumers, and regulators all play important roles in determining how to strike the balance between utility and privacy.

More concretely, methodologies based on shortening sessions, simply remove old query sets assuming that query logs will not be large enough to enable identity disclosure. However, this assumption does not take into account the existence of highly identifying queries. This can be seen on the three-month of query logs published by AOL [40], which does not record the user's AOL screen names, hopping it would benefit academic researchers. Shortly after the publication, some user's names and addresses were identified among the logs. Those logs underscore how much people unintentionally reveal about themselves when they use search engines, and how risky it can be for companies like AOL, Google and Yahoo to compile such data.

Another approach is to remove sets of queries, instead of only the old ones [52], also assuming that the size of the user's logs will not be large enough to allow diffusion of the user identity. In this case, the protection of privacy against highly identifying queries also remain compromised.

A more appropriate approach [26] proposes a solution, based on a threshold cryptography system, that only eliminates highly identifying queries, in real time, without preserving history. They assume that those queries are more likely to refer to identifying or quasi-identifying information. It uses a scheme for encrypting unique queries with a hash that can be decrypted given sufficient examples of the query being used by multiple users. This approach is not free of issues. If an

attacker knows or forces a user to make a query, the attacker can effectively mark all that session.

Also, targeting at unique queries seems to be a overly aggressive technique, as the elimination of a significant number of non-identifying queries becomes a complex and error-prone task. However, this is difficult to avoid due to the fact that the vast majority of queries appear only a limited number of times, as it can be seen at the query traffic analysis conducted on [27]. This analysis was conducted on an hourly basis by matching it against list of queries that have been topically pre-categorized by human editors. In any case, it can be quite challenging to select the proper deletion threshold using this approach.

A report illustrates how top search engines deal with the retention of user's search information [53], also based in suppression methods. In this case, some WSE give to the user the ability to opt out of having WSE retain their search information, including IP address, ID and search query, or delete that information after a certain time. Some apply a personal information filter to remove unique identifiers, names, physical addresses, phone numbers, social security numbers, bank accounts or any other identification data related to the user. It is important to note, that when a user chooses to delete information from their personal search history, it will still remain on the search engine's servers until the minimum retention time defined in their policy, which in some cases is declared to be up to 18 months. Therefore, these data are still exposed to a data breach, even after the user's request for removal.

Nevertheless, the AOL incident reveals the limitations of this approach [41–43]. AOL apologized for its massive search data disclosure, but also declared that the personally identifiable data was removed from these accounts. The existence of quasi-identifiers in the AOL dataset, and the complexity of identifying their combinations, were proven enough to re-identify AOL users via traditional log correlation techniques [40]. Many AOL users could be identified, as vanity searches for name or social network profile, or searches related to city and neighborhood, appear in the search history and it may provide clues to user's identity. Therefore,

this shows us that combining the remaining no-identifying data may be enough to disclose the identities of the individuals in some cases.

More elaborated approaches in this field [20] show how queries, clicks and counts can be published in a perturbed manner that preserves privacy. The conducted perturbation focuses on removing those queries that end up with the user clicking common URLs, considering that those queries may be dependent, therefore, highlighting quasi-identifiers. In addition, they generate keywords from the perturbed data and show that those resemble the ones generated from the original data. However this approach could be deceived, forcing the search engine to publish private data. The authors also state that this approach could be improved using generalization techniques.

Another possibility is the representation of query logs using graph theory [19]. The graph representation is used for query log privacy preservation analysis, defining a heuristic for log anonymization through graph disconnection. In this approach, nodes are seen as user queries. A query is connected to other user queries whenever the intersection of their clicked URLs sets is non empty. Then the system sorts the nodes (queries) on their total number of queries divided by clicked documents. The anonymization process is done by disconnecting the graph. It can be achieved by iteratively suppressing infrequent queries, the queries that return less than k documents. Those queries are most likely to point to individuals (i.e., queries with partial or full target URLs) and therefore are considered more vulnerable, so it is intuitively desirable to suppress them. This approach could benefit from grouping infrequent queries, to obtain higher frequency terms.

As we have seen in the last suppression proposals, a further study on generalization techniques seems that could help to improve anonymization results. In this context, transforming infrequent queries into frequent ones, but other implications may arise, so we will continue studying some proposals that point on that direction.

2.1.1.2 Generalization

Another approach used to provide anonymity is based on the use of generalization between domains relationships and between values that the associated attributes can assume. In general, this approach seeks to make users who performed similar queries, indistinguishable among them.

The concept of minimal generalization was introduced in [54], where they illustrate how k-anonymity can be provided by using generalization and suppression techniques. They introduce the concept of minimal generalization, which captures the property of the release process not to distort the data more than needed to achieve k-anonymity. However, some aspects remain unresolved, the definition of quasi-identifiers and of an appropriate size of k must be addressed. Determining the quality and utility in other settings must be further researched. They also need further investigation of an efficient algorithm, considering specific queries, multiple releases over time, and data updating, which may allow inference attacks.

Due to the dimensionality and unbounded nature of query logs, some authors have proposed a top-down, approaches, using lexical and semantic databases to conduct general-purpose generalizations. As it can be seen in a partition-based approach to anonymize set-valued data that scales linearly with the input size and scores well on an information-loss data quality metric [55]. The proposed algorithms in this case are efficient enough to be applied to non-trivial real-world data, but the information loss results are not as satisfactory in search query logs as when it was applied to other kinds of more structured data.

Another pioneer work on the field of semantic generalization [56], do not distinguish data as sensitive and non-sensitive, but consider all the data as potential sensitive data. The idea is to transform groups of input queries to common conceptual abstractions (e.g. *sailing* and *swimming* to *water sports*), in order to make users who performed similar queries indistinguishable. The main proposed

algorithm finds the optimal solution, however, at a high cost which makes it inapplicable for realistic problems. Then, they propose two greedy heuristics, which find a solution close to the optimal at a minor cost.

For proposals that are based on concept generalization, is very useful the use of knowledge bases, such as WordNet [57]. WordNet is an online lexical database designed for use under program control. English nouns, verbs, adjectives, and adverbs are organized into sets of synonyms, each representing a lexicalized concept. Semantic relations link the synonym sets to determine word definitions.

The main limitations of a generalization approach is that a query could be meaningless in a generic dictionary, but it could be identified as dangerous, according to a more specific dictionary. Therefore, they may need a specific dictionary for each language or sub-domain used on the original data set, which makes the creation and maintenance of these dictionaries an important task for the proper functioning of the proposals, specially in a broad contexts such as query logs. It appears that anonymizing data containing large sets from large domains appears to be a very difficult problem that may require techniques beyond generalization.

2.1.1.3 k-anonymity

To overcome previous limitations, the concept of k-anonymity was proposed [54] which minimizes the risk of record-linkage. They also illustrate how k-anonymity can be provided by using a combination of generalization and suppression techniques.

Conducting a k-anonymity process at the server-side, before releasing the query logs, leads to the release of k-anonymized data-sets. Those data-sets will satisfy the k-anonymity privacy property whenever user data contained within the query logs cannot be distinguished from at least $k-1$ other users — whose data also appear in the release. In other words, no individual can be re-identified with probability exceeding $\frac{1}{k}$ through linking attacks alone.

In a more recent paper [58], the author describes how the techniques developed for protecting data have evolved in the years. It starts by providing an overview of the first privacy definitions (k-anonymity, l-diversity, t-closeness and their extensions) aimed at ensuring proper data protection against identity and attribute disclosures. It also shows the impact on privacy when considering multiple releases of the same data or dynamic data collections, fine-grained privacy definitions, generic privacy constraints, and the external knowledge that a potential adversary may exploit for inferring sensitive information. Those are the main shortcomings of the k-anonymity based approach. They also briefly present the concept of differential privacy, as an alternative privacy definition, which we will study in detail later.

Next step in the evolution of proposals were provable anonymization methods based on *Statistical Disclosure Control* (*SDC*) techniques [33]. Those methods, must be conducted to guarantee bounded disclosure risks [59]. Their goal is to transform query records into anonymous logs while retaining the maximum level of information and reducing the amount of query deletion.

We can see an example in [60], where an anonymization model is presented. This model extends the notion of k-anonymity and provides metrics for user similarity and query similarity. The inherent utility and privacy trade-off was analyzed, and experimental results show that with this model it is possible to achieve high user similarity in the anonymized data quite efficiently. However the suitability of the method remains untested in different important applications, such as query log mining.

Also, [61] introduces an approach to anonymize search query logs by means of microaggregation to protect the user privacy. In this approach there is also an attempt to provide a high degree of privacy ensuring k-anonymity on the data, without having to completely eliminate any record, to allow the search logs to retain its usefulness. However disclosure risk and information lose of the data remains untested. Besides, a better aggregation of query terms by means of semantic generalization could be obtained using the WordNet lexical database.

Another subset of approaches try to use logs of similar queries to group users, and later their queries are rewritten by a prototype query, so that they become indistinguishable. We proceed to study those approaches in more detail below.

In [62] a new scheme that generates distorted user queries from a semantic point of view in order to preserve the usefulness of user profiles is presented. The new queries, created by analyzing those already performed by the user, introduce a configurable degree of information distortion proportional to the desired level of privacy. The most CPU-demanding task in this scheme is the retrieval of concepts from the background ontologies. Due to the lower-efficient access method and the larger size of the *ODP* repository, the analysis of a query may take up to an average of 1 500 ms, and this is the main disadvantage of the proposal.

An improved method based on micro-aggregation [63], uses search results to increase the semantic interpretation of the query logs, providing more data reliability than previous methods. The improvement is solely based on the new similarity function, but as they state, using a clustering heuristic such as *Maximum Distance Average Vector* (*MDAV*) can improve the achieved results in those techniques.

The usage of *MDAV* was introduced in [64], which based on semantic micro-aggregation, enable the publication of privacy-preserved query logs. Query logs are clustered into groups, using *ODP* categories, of size k (data partition) and replaced by the cluster centroid (data anonymization) using *MDAV*, which establish the base of their query anonymization proposal. Thanks to that, the semantics and distribution of original queries was preserved, while disclosure risk was kept at a reasonable level.

Another approximation using *MDAV* was presented on [65], that defines an anonymization method for set-valued data based on semantic micro-aggregation by extracting their conceptualisations from an ontology. This enables to aggregate set-valued data from a semantic perspective by means of an adaptation of the *MDAV* algorithm. In this case, the synthetic records that semantically match the cluster centroids are generated by randomly picking values from different records of the input data set.

Also using micro-aggregation, we found another approach for query logs anonymization [66]. In this case the original queries were also replaced by the centroid. The novelty of this approach is that the other parts of the log are aggregated by using the arithmetic mean for the rank and the timestamp, and generalizing the user selected *URL* to the right-most common part, i.e. the sub-domain. This proposal seeks to achieve full k-anonymity of the user in the query log.

Up to this point we have seen that users and queries are conserved, although queries are transformed to reduce the risk of disclosure. However, alternative approaches have been proposed to generate fake messages, and mix them together with the legitimate ones to make more difficult to relate sensitive information. As an example, [67] presents a scheme that obfuscates the real user's profile, generating fake messages together with legitimate ones. The fake messages use terms semantically correlated with user posts to distort and, hence, hide the real profile. This scheme effectively distorts user profiles, producing uniform (i.e. balanced) profiles that hardly characterize any user. This scheme could face limitations against ad hoc profiling methods specially designed to detect fake messages. So custom systems that recognize and omit fake messages could be developed using trained classifiers or defining specific detection rules, so the protection of this proposal could be defused.

As many infrequent queries can be seen as refinements of a more general frequent query, a different approach [68] tries to mask the infrequent queries using the general ones. It presents a semantic approach applied in three steps: query concept mining, automatic query expansion, and affinity assessment of expanded queries. Doing so, the approach is able to mask identifying queries while retaining a substantial amount of highly infrequent queries, achieving levels of privacy comparable to those of plain k-anonymity while at the same time reducing the data losses.

Statistical disclosure control, so far, seems the more suitable approach for our purpose. It could be accomplished trough different anonymization techniques and statistical criteria, being k-anonymity the most widely accepted technique. Event

though k-anonymity seems very promising, all the approaches based on the generalisation of original values suffer from a high information loss, derived from the reduced granularity of the output values. Moreover, its application is limited solely to finite sets of data. As in the context we are studying the data is generated continuously, data utility will be one of our concerns. We need to look further for a more appropriate approach for our case.

2.1.1.4 Differential privacy

Initially described as a solution, which, intuitively, captures the increased risk to one's privacy incurred by participating in a given dataset [69]. However, an auxiliary information generator with information about someone not even in the database can cause a privacy breach to this person. In order to sidestep this issue they part from absolute guarantees about disclosures to relative ones: any given disclosure will be, within a small multiplicative factor. As a consequence, there is a nominally increased risk to the individual in participating, and only nominal gain to be had by concealing or misrepresenting one's data.

Using this idea, interactive scenarios of the same approach do also exist [70], to provide safe interactive access to search logs, instead of releasing them. In those methods, who accesses the data is just allowed to make some queries, accessing partial information of the data-set. However, if these queries are intelligently conducted, they may end up revealing information from the original user. For that reason, semantic policies are used to infer the higher levels of information that can be mined from a data-set based on the fields accessed by a researcher. The accessed fields are then used to build research profile(s) that guide the amount of privacy to be enforced using differential privacy, and if the fixed privacy limit is surpassed, then the system stops responding, which is the first limitation of this proposal. A second limitation is that the raw search logs cannot be given or sold to a third party, so they are limited to conduct research over the logs using the provided tool.

Since the protected output may still preserve some statistics (e.g., query suggestions, spelling corrections and query classification), extended proposals [20] aim at similar systems, but returning only some statistics to further limiting the risk of information disclosure.

Authors in [71] propose a technique which seeks to greedily select representative samples in query logs with high utility and provide privacy guarantee by prior estimation of n-word phrase utility. In the estimation, they utilize click entropy to measure the click distribution of the same query. In this way, samples with high utility are selected to become the representative records in each cluster without sacrificing privacy while reducing redundancy, which can help to achieve the objective of leaking less privacy and releasing more useful information.

We can also find other proposals using differential privacy on query logs to guarantee high levels of privacy [72]. In this case, they also provide a proof for why the user IDs in each individual search records can not be released in order to achieve such differential privacy. They also illustrate the privacy-utility trade-off in query log releasing process. Regarding to this, the proposal was too strict to use in a WSE environment and some settings need to be relaxed to maintain data utility of the released query log. A similar framework [73] was introduced to anonymize query logs by differential privacy. The framework is empirically evaluated and experiments show that the Web search algorithms using the anonymized logs do not perform significantly different from those using the original logs. They also suggest using more relaxed settings in a WSE context, to maintain high data utility. Both cases pose the addition of Laplacian noise to the logs, to preserve privacy. However, the noise added is highly related with the loss of data utility. The more noise is added, the more data utility gets reduced.

In general, differential privacy methods, still use some kind of data perturbation. Therefore all of them suffer from unmitigated risks, such as leakage in queries and URL, as well as noise addition.

2.1.2 Data-stream input

Data-streams can be considered as one of the main sources of what is called big data. Therefore, it has been the target of some research in recent years. However few proposals has been directly applied to WSE query logs viewed as a data-stream, but this kind of approach could be relevant for our research because it allows to treat data partially. Considering that data-stream usually contains huge volumes of data and could be virtually infinite, techniques that do not need the full data set are definitely deserve to be considered.

The storage, querying and mining of such data is a resource intensive task, and maintaining the privacy is challenging. Previous reviewed *statistical disclosure control* (*SDC*) or privacy-preserving techniques, usually cannot be applied directly in a data-stream environment.

We need schemes that consider data as an unlimited stream and that are able to process information as soon as it is generated. A preliminary review of some proposals in this regard [74] points that this kind of systems do not need to store all the data to start dealing with it, and also are able to generate data output with a minimum delay. It also stands out that due to the huge amounts of data, it is important to design efficient techniques that can have only one look or less over the incoming data-stream, and this should be accompanied with acceptable result accuracy. Some of the reviewed proposals include an interactive mining environment to satisfy user requirements. However, mining data streams is a highly application oriented field and it is hard to generalize in broad contexts such as query logs. Other approach, is to define a pre-processing step that can guarantee quality of the results. However in this case data pre-processing is a resource intensive step. Therefore, mining massive data streams that have privacy protection in a network environment is a challenging task.

Also [75] presents a discussion on open challenges for data stream mining. The identified challenges cover the full cycle of knowledge discovery and involve such

problems as: protecting data privacy, dealing with legacy systems, handling incomplete and delayed information, analysis of complex data, and evaluation of stream mining algorithms. They conclude that in many cases it would be beneficial to step aside from building upon existing offline approaches, and start blank considering what is required in the stream setting.

Below we present the main data-stream input proposals, grouped according to the data processing technique they use: Rank swapping, differential privacy, k-anonymity and probabilistic k-anonymity.

2.1.2.1 Rank swapping

Rank swapping is a well known method for *Statistical Disclosure Control*, which ranks the original data and then randomly exchanges the values between records that are in near positions on that rank.

The method was first described for ordinal variables in [76]. Although initial ideas associated to swapping data exist in other previous areas [77] where it is presented as an alternative for the release of micro-data from a statistical database. Their results show that the resulting data will be close to the original one in terms of the desired statistics, but preserving the confidentiality of the original data. It also becomes clear that approximate data-swapping can be applied practically to large categorical databases with a range of statistical information that is to be preserved. However, the study of approximate data-swapping is not complete, so other methods must be sought.

Although originally described only for ordinal variables, we can also find other methods proposed for any numerical variable [78]. In this case, values of variable V_i are ranked in ascending order. Then, each ranked value of V_i is swapped with another ranked value randomly chosen within a restricted range (e.g., the rank of two swapped values cannot differ by more than p percent of the total number of records).

Detailed results could be found in [79], showing that rank swapping techniques outperform the best micro-aggregation techniques up to this point. In fact, the best performer is using a p around 15 percent.

All their proposals use structured data, to facilitate the arrangement of data in a ranking using the value of an attribute. The implementation of masking random noise used on the tests, uses a simple algorithm for uni-variate Gaussian noise generation, and further tests with multivariate Gaussian noise is required. The results show that information loss and number of re-identifications are highly dependent on the set of variables and the number of categories in each variable. All this makes these proposals less applicable to our goal.

A different approach [80] studies the application of rank swapping to numerical data streams, composed by records referring to individuals or entities which are generated as streams. In this case, the output can be stored for future analysis or even analyzed in (close to) real time, without risking the disclosure of sensitive information from the data. This method was studied in terms of information loss and could be used as a starting point for considering rank swapping a candidate to be applied in streaming data.

As seen in this section, the traditional approaches consist in using one variable at a time for rank swapping, however at least a generalization of the method using multivariate ranking/mapping functions should be defined as a first step to use those methods over more complex data. However, this is not a trivial undertaking, as more complex ranking/mapping could lead to very poor fits. Finally, no application of ranking algorithms was found using query logs or other types of unstructured data.

2.1.2.2 Differential privacy

As seen in 2.1.1.4, the concept of differential privacy was defined in [69]. The same approach can also be applied to anonymize data-stream environments. In a sample application [81], there is no release of the original query, but a synthetic

one, obtained using semantic similarity. In this way, they preserve the cardinality of the query logs, and also drive the distortion consistently with the semantics of the queries. That is, the queries used as replacements of the original ones are probabilistically chosen according to their semantic similarity and the desired level of protection. This contributes to preserve the utility of the protected query logs. However, the proposal remains untested using structured knowledge sources other than *Word-Net*, such as *ODP*, *YAGO* or *DBPedia*, which may offer more detailed and finer grained taxonomies that could also be used to increase the accuracy. The lack of structure of query logs, combined with new terms which may not be present into the semantic database, could represent a challenge for this approach.

Data micro-aggregation, which consists on grouping similar records/logs together and replacing them by a representative value, can be used to decrease the sensitivity of the differentially private mechanism proportionally to the size of the data aggregation. The main challenge in this case, would be to adapt the micro-aggregation algorithm, to the lack of structure of query logs.

Another limitation using differential privacy in a streaming environment is to maintain a fixed privacy level. It is possible that no more data can be published in order to preserve user's privacy.

2.1.2.3 k-anonymity

As the previous methods, k-anonymity was also firstly proposed only for static data sets. All the methods for static data anonymization cannot be directly applied to anonymizing data streams, however, below we will follow the evolution of the proposals regarding to k-anonymity in more complex environments.

The use of k-anonymity in high dimensional data, such as query logs, was studied in [82]. This approach is relevant since k-anonymity was designed for structured databases with a limited number of attributes. They discuss the effects of the curse of high dimensionality on privacy preserving data mining algorithms. As more attributes are added to the data, the possibility of inference attacks increases, as

it is proportional to the number of possible combinations of attributes. Also in their proposal, when the data contains a large number of attributes which may be considered quasi-identifiers, it becomes difficult to anonymize the data without an unacceptably high amount of information loss, and performance is severely hampered with high dimensional data. This is because an exponential number of combinations of dimensions can be used to make precise inference attacks. More appropriate approaches must be sought.

A proposal that tries to improve the performance [83], presents a one-pass algorithm, using weak clustering based data streams k-anonymity method for data publishing. This method first measures information loss of each anonymous cluster, then it considers data privacy and data protection achieved according to continuous classification attributes of data stream. It uses little processing time and memory for each tuple of data steam, and both privacy preserving and utility of anonymous data are considered. However, they found some concept drift, which needs further research. Also some aspects of data anonymity remain untested.

Deeper anonymization schemes were proposed [84], integrating two approaches, data stream management and privacy protection. They propose a method called *SKY* (*Stream K-anonymitY*) to continuously facilitate 6-constraint k-anonymity on streaming data for privacy protection.

A similar approach can be found in [85], where they study a framework named *KIDS* (K-anonymization Data Stream based on sliding window) to solve this problem by continuously k-anonymity on the sliding window. It adopts a top-down specialization tree (*TDS-Tree*) for completing the k-anonymization, and adjust the *TDS-Tree* with each update of sliding window. Regarding to the results, it protects privacy of data stream and considers the distribute density of data in data stream, thereby improve usefulness of data over previous methods.

Other alternatives using randomization exist [86]. They tackled the problem of continuous privacy preserving publishing of data streams, by using an approach which considers both the distribution of the data entries to be published and the statistical distribution of the data stream. In some proposed applications, it

is required to monitor certain statistics or maintain a classification model on a published data stream.

However, those methods present some inconvenience. First, a delay, as they need to wait for new tuples to cluster the data stream. Second, it is hard to determine how the data stream could be published so that the privacy preserving requirement is honored and the statistics and classification model can be mined as accurately as possible.

A different clustering proposal [87] defines an algorithm to publish the tuples without violating the anonymization principles and keep the information loss as low as possible. They employ a clustering strategy with tuples at its core and a cluster reusing strategy. The reuse constraint of the stored k-anonymized clusters also sets a threshold to the size of the k-anonymized cluster set, which is vital to the linear time and space complexity of the proposed algorithms. However some issues remain unsolved. The clustering strategy may publish a newly arrived tuple early before its time limit. So, it will cost extra information loss if the new tuple can be published with lower information loss by a cluster created later. Moreover, it is possible that the additional information loss caused by substituting another tuple for the new tuple is greater than the information loss reduction acquired by deferring its publication. However, the prediction process may be time-consuming.

We found another cluster-based proposal [88] based on a *Fast Anonymizing Algorithm for Numerical Streaming daTa* (*FAANST*), which can anonymize numerical streaming data quite fast, while providing a moderate data loss. It uses a clustering algorithm, k-means, which takes advantage of the concept of a centroid. However, this concept can not be defined on categorical values, which arises the main problem of this proposal, that it can only anonymize numerical data. Some additional work to adapt this approach to support categorical attributes, could be useful to our research.

Related to multivariate data-sets, an interesting approach [89] propose a *Fast Data-oriented Microaggregation algorithm* (*FDM*) that efficiently anonymizes large multivariate numerical data-sets for multiple successive values of k. The *FDM* can

save a significant time and resource in protecting large numerical data-sets. It is also shown that the method usually achieves a better trade-off between disclosure risk and information loss measures. However, other data types different from the numerical ones, remains untested.

Up to this point, presented methods are mainly based on clustering incoming data streams. All of them need to wait for new tuples in order to build the anonymized clusters. In order to avoid the problems inherent to accumulation-based methods, the authors of [90], present a delay-free anonymization framework for preserving the privacy of electronic health data streams. Input streams are anonymized immediately with counterfeit values. However, data utility of the anonymization results remains low and the counterfeit values unmanaged. Also data throughput of the proposal is low to process high volumes of data.

Despite being interesting, an important issue of traditional k-anonymity approaches is the difficulty of using unstructured streams of data while satisfying the aforementioned privacy properties. Some of them are not fast enough, which poses an additional problem to WSEs requiring, moreover, real-time processing. Also, enforcing strict k-anonymity implies variability loss and therefore quality loss. This is especially serious in a scenario with informed intruders, who know the values of some confidential attributes as these attributes can be viewed as additional quasi-identifiers. The more quasi-identifier attributes, the more data quality loss is caused by k-anonymity. So we continue to seek alternative strategies that allow us to overcome those limitations.

2.1.2.4 Probabilistic k-anonymity

The concept of probabilistic k-anonymity was introduced on [91], which relaxes the indistinguishability requirement of k-anonymity. More precisely, it only requires that the probability of re-identification is maintained, with regard to the case of k-anonymity. Like standard k-anonymity, probabilistic k-anonymity guarantees that the probability of correct re-identification is at most $\frac{1}{k}$, but without explicitly

requiring that the quasi-identifier attributes take identical values within each group of k records.

Their work shows that, for a fixed re-identification probability $\frac{1}{k}$, the probabilistic k-anonymity methods are much more quality-preserving than standard k-anonymity enforcement.

An evolution of the concept can be seen in [92]. In this case they propose a system that dynamically anonymizes data to compile privacy-preserving query logs that may be monetized. The proposed scheme has been designed to process queries as fast as possible, to be able to process all the queries received by a *WSE*. It also takes into account the degree of privacy of the individuals.

Probabilistic k-anonymity does not have the same limitations as those we have seen so far. By relaxing the requirement of indistinguishability, a better use of the data may be accomplished. Another improvement that brings the application of these methods to query logs is that the original queries can be released, instead of having them replaced with some synthetic ones, just as with many of the previous described techniques.

Proposals in this section point in the same direction. From all the studied alternatives, this seems the one which better suits our needs. On the negative side, given the continuous generalization of unstructured dataset elements, a certain imprecision is added to the generated profiles. Existing limitations in the related literature [92] are found in terms of classification methods, which are very basic. Hence, resulting in a low category count that leads to high degrees of data utility loss.

2.2 Survey on Client Side Proposals

One may argue that WSEs have no motivation to protect the privacy of their user. Indeed, user may be seen as the only interested party responsible to protect data privacy. Under this assumption, we find some protection approaches which do not

expect any collaboration between WSEs and user. Such approaches can be classified in two main categories: i) obfuscation techniques and ii) anonymous channels. Obfuscation techniques generate noise to distort the user's profile managed by the WSEs. Anonymous channels assume an infrastructure between user and WSE to handle the profiling of activities. The use of client side techniques are assumed to generate non-realistic profiles that may have an adverse effect on the services provided by WSEs.

2.2.1 Obfuscation Techniques

Early techniques assume the introduction of random queries (e.g., fake queries), in order to obscure user's profile. Random queries must be indistinguishable from the real queries. This property is known as unobservability. Representative solutions based on obfuscation techniques can be classified according to the number of users that participate in the protocol. We found standalone solutions and distributed solutions. Standalone solutions assume individual user handling their own privacy in front of the WSE. Distributed solutions assume groups of users working together to protect the privacy of each user. Next, we provide some examples for each category.

2.2.1.1 Standalone Systems

These schemes generate synthetic queries that are used to hide the real queries of the user [11, 62, 93–99]. Synthetic queries are submitted together with the real queries, obfuscating the profiles that the WSE owns for each user. If the synthetic queries are in some way semantically related to the user's queries, the obfuscated profile will still be usable, i.e., the WSE will be able to personalize the user's results. When the synthetic queries are semantically unrelated to the user's queries, the profile will be heterogenous and the personalization will be less accurate. This does not mean that one alternative is better than the other, since a user may have different preferences regarding of the trade-off between privacy and

utility. However, some works show that it is possible to distinguish real queries from synthetic queries [97, 100–102]. These works rely on the idea that machine-generated queries do not have the same features as human-generated queries.

2.2.1.2 Distributed Systems

These schemes require the collaboration of a group of users that work in partnership to protect their privacy, i.e., they hide their actions within the actions of many others [103–109]. Typically, these schemes put a user into a large group where they submit requests on behalf of other members. Users exchange their queries. Personalization is only possible if the members of the group share the same interests [3]. In some proposals [103–105], there is a central node that poses a bottleneck in the overall system performance. In other cases, one type of path [103, 106–109] is created to submit the query or a group of users must be created [103–105]. In both cases, a significant delay is introduced [3].

2.2.2 Anonymous Channels

Proposals under this category use anonymous infrastructures [110, 111] in order to send user's queries to the WSE. By concealing user's identity associated to the queries, WSEs are assumed to be unable to profile the user. However, this may affect the quality of the service offered by the WSEs to the user.

Chaum's mix networks [112] are representative cases of solutions under the category of anonymous channels. Messages pass through several nodes. Each node disassociates the input messages from the output messages, by means of cryptography [110, 111]. Evolved techniques assume the use of proxies [113], relying connections (e.g., queries) from user to the recipient (e.g., the WSE). The key concept is that the proxy delivers the messages but does not disclose the source (e.g., the user' identity). DuckDuckGo[3], Start Page[4] and Yippy[5] are some significant

[3]https://duckduckgo.com/
[4]https://www.startpage.com/
[5]https://www.yippy.com/

examples using proxy-like infrastructures. By using these solutions, user transfer their trust from WSEs to the proxies (i.e., user must assume that proxies do not monitor or log their traffic).

Web MIXes [114] provides anonymous and unobservable real-time Internet access. It incorporates an authentication mechanism in order to prevent flood attacks. Additionally, it includes a feedback system with an interface that informs user about their current level of protection. However, some flaws in their authentication process may allow external attackers to perform replay attacks [115]. The synchronous nature of Web MIXes may also generate problems when dealing with asynchronous TCP/IP networks [116].

The use of onion routing [117] to establish anonymous channels under the context of queries and WSEs has also been proposed in the literature [118]. General purpose plugins, and modified web-browsers[6] using the Tor Project [119], are user-friendly solutions based on the onion routing paradigm. Similarly, the Invisible Internet Project (I2P) [120] builds an anonymous network layer designed to be used for anonymous communication. Nonetheless, several weaknesses have been reported [121], and Tor does not attempt to offer security against passive global adversaries [111].

2.3 Survey on Collaborative WSE-Client Proposals

Solutions under this category assume that user and WSE work together in order to protect user's privacy. Next, we report solutions under this category in three main groups: i) Private Information Retrieval; ii) Platform for Privacy Preferences (P3P); and iii) Context-based Retrieval.

[6]https://gitweb.torproject.org/tor-browser.git/

2.3.1 Private Information Retrieval

Private Information Retrieval (PIR) schemes [122–125] enable user to obtain information from a database privately, i.e., the server cannot know what information was retrieved. Through a PIR scheme, user can search documents stored in the database and recover those of their interest. The problem of submitting a query to a WSE while preserving the user's privacy is equivalent to the PIR problem. However, PIR schemes suffer from two practical problems that make them not appropriate for WSEs [104]: PIR schemes are not suitable for large databases, and user is assumed to know the precise location of the records to be recovered.

2.3.2 Platform for Privacy Preferences (P3P)

The Platform for Privacy Preferences (P3P) [126, 127] was created by the World Wide Web Consortium (W3C) with the objective of making easier for user to obtain information about the privacy policies of the sites that they visit. P3P is a framework through which user can automate the protection of their privacy. They can define their privacy preferences and, when a website does not conform to these preferences, then P3P-enabled browsers may alert the user and even take pre-established actions (e.g., deny access to cookies). The Do-Not-Track initiative [128] is a policy-based P3P system in which HTTP headers request web applications not to track the user. The web application must be P3P-complaint in order to be effective. It has been studied in several works [129–131] and standardized by W3C. However, it is considered as an obsolete protocol nowadays. In fact, P3P-like solutions have been criticized due to the impact that governmental laws may have over the user [132], the lack of follow-up from websites w.r.t. privacy-protection mandates in their legal jurisdictions (e.g., compliance difficulties of websites to enforce their own privacy policies) [133], and low number of potential adopters [134].

2.3.3 Context-based Retrieval

Context-based retrieval proposals aim at storing user profiles (e.g., search history) on the client's machine. This information allows to obtain user's interests and re-rank search results according to them. WSE and user participate together in the searching process in order to obtain the final results, i.e., the WSE receives the query and returns the results. Then, these results are re-ranked at the client-side. The User-Centered Adaptive Information Retrieval (UCAIR) project [135] collects and exploits available user context from submitted queries and clicked results. Similar schemes allows user to choose the content and degree of details of their profiles exposed to the WSE [13, 136, 137]. In the end, user determine the profile content that is revealed to the WSE when a query is submitted. The adjustment of parameters associated to the stored profiles is possible, in order to improve the quality of the results. Potential disadvantages of these proposals relate to performance and effectiveness limitations of results ranked at the client (i.e., much less effective than ranking the results at the server side) [135]. Moreover, it is expected that WSEs can still profile the user after several executions of the approach.

2.4 Final thoughts

Within this broad classification that we have studied extensively, three main types of proposals were found, regarding to the actor who applies them: server-side, client-side or collaborative proposals.

We assumed that for monetization purposes the goal of WSE is to preserve as much data utility as possible, the feasibility of client side approaches seems limited in this context. As already discussed, introduction of fake queries, delayed response time as well as mixed user profiles, generate drawbacks in result's data utility.

Regarding to collaborative proposals between WSE and clients, we found distinct approaches. Private Information Retrieval (PIR) schemes, that have turned out

not suitable for large databases, like the ones on a WSE environment. Platform for Privacy Preferences (P3P) solutions have been criticized due to the impact that governmental laws may have over the user, the lack of follow-up and low number of potential adopters. Finally Context-based Retrieval showed potential disadvantages related to performance and effectiveness limitations of results ranked at the client. Additionally, it is expected that WSEs can still profile the user after several interactions. Therefore, a certain degree of trust must be deposited in the server, even when achieving a reduced data utility, as shown.

Therefore, server side approaches seem more appropriate to maximise both privacy and data utility over large amount of data. Such approaches use two great types of methods to protect data, those who use perturbative techniques and those who use nonperturbative techniques.

With perturbative approaches, the data-set is distorted before publication. As a result, unique combinations in the original data may disappear and new combinations may appear in the perturbed data. Such changes may be beneficial for preserving confidentiality, but may have an adverse effect in relation to data utility.

On the other hand, nonperturbative methods do not alter original data, which is more appropriate to preserve data utility. These methods use techniques such as global recoding, partial suppression and sampling, to preserve user confidentiality.

From the point of view of their data input, server side anonymization methods fall into the following two broad categories: fixed-length input and data-stream input. We will analyze them in the sequel.

2.4.1 Fixed-length input

We found some works that try to anonymize textual query logs, but that do not consider the dynamic nature of those sources of data, and treat them as a fixed-length input. Fixed-length input approaches, work well only with static data. This implies that, if they were used to anonymize query logs, these sources of data must be “closed” before applying any procedure.

According to this, and depending on their size, two main options are considered: i) the query logs to be anonymized contain all the queries received from time 0 until now; or ii) the query logs to be anonymized correspond just to a short and limited period of time.

In the first option, with all the queries received from time 0, the anonymization method in use will deal with a huge quantity of data; even worse, query logs are expected to grow daily and, hence, each anonymization process will be required to process bigger amounts of data. In order to show the relevance of this problem, it has to be noted that usual anonymization methods that provide k-anonymity in databases have shown significant limitations when dealing with large quantities of data. For example, well-known *SDC* techniques based on generalization or micro-aggregation suffer from a very high cost when performing the partition of the database, in this way, authors have reported quadratic costs in this steps that clearly disqualifies these techniques for the volume of information that is considered in this case [138]. Being more specific, the authors in [139] apply semantic micro-aggregation to anonymize *WSE* query logs and their conclusion is that these techniques do not fit well with large sets of data that require a continuous update of the anonymization due to new data being added continuously.

Regarding the second case, anonymizing a short period of time, an organization that builds query logs can work with "limited" sets of data by means of classifying received search queries by week, month, trimester or semester; then, at the end of the chosen period of time, just the corresponding and limited query log is anonymized and disclosed. This option clearly alleviates the cost required by the anonymization method in use; however, it only focus on a specific window of time, which implies that a lot of queries are discarded and, hence, the accuracy of the anonymized outputs is expected to be jeopardized due to the loss of information.

In conclusion, "closing" query logs, which are a dynamic source of data that is continuously growing with new search queries issued by the *WSE*'s user, is contrary to their real nature and it has been acknowledged in the literature that it is an ineffective strategy. It may be interesting from a research point of view, but

hard to translate to a real environment. As a consequence, we need to focus on methods for real-time processing that may be applied to the considered scenario, using data-streams.

2.4.2 Data-stream input

As we have seen to this point, *WSE* query logs produce a huge volume of data which needs to be dynamically treated and that grows continuously. This fact makes traditional methods not practical for their application in this context, and only data-stream based methods seem to make sense.

If we look for proposals with this approach on the literature, we have found some custom designed methods to privacy-preserve data on a streaming context. However, the fast proposals focus only on structured information. More specifically, the considered data input is mainly numerical, which is the main limitation when considering its possible application to the anonymization of *WSE* query logs. As it has been shown, in query logs of the *WSE*s it is usual to find unstructured data made of both textual and numerical data referring to searched keywords.

The proposals closer to our needs, are the ones that use methods of probabilistic k-anonymity, but unfortunately, these methods are in an early stage of development and therefore we should work on improving them. If we could improve them, the result could be an interesting approach to anonymize query logs.

In addition, we want to consider the utility of the resulting anonymized data. This is a critical aspect to facilitate the exploitation of data to third parties. Therefore, a study of the differences in the utility of the data is essential, and we have not found it in the previous described methods. This fact, in conjunction with the limitations that we have exposed shows that there is still room in the literature for new schemes specially designed to anonymize dynamic query logs while maximizing the data utility.

Non-perturbative approaches seem more suitable to facilitate the exploitation of data by third parties with the adequate protection of sensible user information, owing to the fact that they achieve a high level of data utility.

Therefore, our proposal will focus on suitable non-perturbative server side methods to deal with a data-stream input in real time. The use of probabilistic k-anonymity, seems appropriate to bound disclosure risk of personally identifiable user attributes. Additionally, our solution should be able to handle unstructured data. It should provide a fast probabilistic method to blend streams of queries with high similarity to those requiring protection, but coming from different users.

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

Chapter 3

Working at the WSE side to generate privacy-preserving user profiles

3.1 Introduction

The popularity of Web Search Engines (WSEs) has grown with the number of websites present on the Internet. According to the Netcraft January 2020 Web Server Survey, there exist 1 295 973 827 (over 1.2 billion) websites, 249 618 033 unique domains, and 9 576 845 web-facing computers [140]. Therefore, it can be assumed that WSEs will continue to be essential to surf the Web.

When a person submits a query to a WSE, it looks for the requested information among its indexed web pages, but it also stores the submitted query (i.e., the keywords) and some metadata (e.g., date of the query, some identifiers of the query issuer, which specific search result was selected by the issuer, etc.). As a result, everyone who uses a WSE is disclosing personal data, such as personal characteristics and preferences, and enabling WSEs to compile those query logs.

Query logs make up a set of unstructured data, which is generated as a continuous data stream. Although a fast process could protect query logs prior to their publication, there is no absolute guarantee of anonymity, as the combination of modified data may disclose enough information to re-identify some users [19, 141]. As an example, there is one well-known case, the AOL scandal, in which around 36 million queries performed by AOL's costumers were publicly disclosed. Although records were previously de-identified, it was possible to identify some users from the disclosed query logs and other sensitive information was exposed [40]. This case ended up with an important damage to AOL users' privacy and to AOL itself, with several lawsuits against the company [42].

Therefore, in order to get viable data monetization, better tools capable of modifying query logs by limiting the privacy disclosure risk but preserving as much data utility as possible should be provided. More specifically, in this chapter, we propose using a server side software capable of processing queries, as unstructured data, in real time and building anonymized query logs that still retain enough data utility to allow its monetization. As a result, WSEs may then offer the protected query logs to external organizations for data monetization purposes while keeping the real query logs in a safe place; otherwise, WSEs may also decide to only keep protected logs, getting in turn a lower risk of information disclosure in case of a direct attack.

The rest of this chapter is organized as follows: in Section 3.2, we define the requirements of the system. In Section 3.3, we describe in detail the proposal, used to implement the system. In Section 3.4, we analyze the results. In Section 3.5, we highlight the conclusions we have reached.

3.2 Requirements

Our proposal, in a nutshell, is based on a server-side architecture that enables WSEs to anonymize query logs in a streaming environment. The outputs of this system are two: i) a real-time stream of anonymized logs; and ii) a database of

user profiles. The main target is to allow WSEs to sell or release both data sets to interested third parties without threatening the privacy of the individuals who have filled the query logs with their issued search queries. In order to achieve that, the resulting outputs must fulfill certain requirements that are next detailed.

3.2.1 Privacy requirements

The main requirement for the proposed system is that it must preserve the privacy of the individuals who contribute to the WSE query logs.

Privacy could be achieved by means of query de-identification, following one of the following approaches:

- **Full de-identification**: it is achieved when all identifiers, direct and indirect, are removed and there is no reasonable basis to believe that remaining information can be used to identify an individual.
- **Partial de-identification**: it is achieved when only direct identifiers are removed (indirect ones remain).
- **Statistical de-identification**: it is achieved by maintaining a trade-off between keeping as much useful data as possible while guaranteeing statistically acceptable data privacy.

Direct identifiers (e.g., full name, national id, etc.) can be easily removed from query logs. However, the textual content of search queries may contain any possible value of any domain; this is very likely to produce cases in which may be very difficult to distinguish identifying queries from innocuous ones. As a result, in the considered scenario full de-identification requirement cannot be guaranteed.

Partial de-identification is easier to achieve and, additionally, more data utility may also be preserved. However, it is prone to record-linkage attacks [54] that would allow to re-identify users by means of certain apparently innocuous queries.

Statistical de-identification seems the more suitable approach. It is obtained trough certain statistical criteria and anonymization techniques, being k-anonymity and its extension l-diversity the two most widely accepted models. Those models were proposed for structured data [85, 86, 142]. However, the current proposal seeks to prove its usefulness also when anonymizing unstructured data streams.

3.2.2 Functional requirements

To enable the use of current proposal in a real environment, it must also fulfill a set of functional requirements:

- **Scalability**: It is the capability of a system to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth [143]. Specifically we want to achieve load scalability, that is the ability to easily expand or contract to accommodate heavier or lighter loads. Scalability methods fall into two broad categories [144]: i) *Horizontal scalability* which is related to the ability of a system to add more working nodes, such as a new computer; and ii) *Vertical scalability* which is related to the ability of adding resources to a single node in a system, typically involving the addition of CPUs or memory. Both approaches have their own trade-offs, and the proposed architecture should take advantage of all the available resources. Configuring an existing idle system has always been the less expensive alternative, regardless of the approach. So the proposed architecture should be designed to take advantage of this fact and, therefore, it should use the existing WSE infrastructure.

- **Processing speed**: In order to minimize delays generating anonymized data stream, a high processing speed should be a main requirement. It's also fundamental in order to being able to process all requests received by the WSE. A WSE receives every second thousands of queries. For example, Google is processing in average 40000 queries per second [145]. This fact implies that

the new proposal should be able to meet similar speed requirements as the ones met by WSEs.

- **Resource consumption**: As an additional requirement, the resource consumption of the system must be low, in order to facilitate its inclusion in an existing WSE, minimizing overhead cost. Even though the system is supposed to scale with added resources, it is important to not increase unreasonably current WSEs resource consumption.

- **Transparency**: To ease the integration of the proposed system in existing WSEs architectures, it must be transparent. New modules can hid internal details, making them invisible for the main architecture, i.e. encapsulated. Mainly, they should adhere to previous external interfaces without changing them, while changing its internal behavior, i.e. generating anonymized query logs with the same structure than original logs. The main purpose of providing a transparent system is to avoid changing any existing part of the WSE to be integrated with.

- **Modularity**: As explained above, the proposed solution should be easily integrable in an existing WSE. According to this, functional scalability should be achieved, i.e. being able to enhance the system by adding new functionality at minimal cost. To do achieve this, the proposed system should be designed to be modular and to have low coupling and high cohesion.

3.2.3 Utility requirements

Although full de-identification is desirable from a privacy point of view, ideally, preserving the privacy of the individuals should be compatible with allowing WSEs to sell non-sensitive user information to third parties. Logically, the economical value of these privacy-safe data will directly depend on its remaining quality from the utility point of view. As a result, there is a clear trade-off between anonymizing logs and keeping them useful to extract information through data mining processes. Therefore, the main challenge related to data utility is to anonymize sensitive user

data, removing as few information as possible in order to have enough interesting information to be analyzed.

To achieve this, the proposed system aims to build fake logs and user profiles which should retain users' interests (maximizing the data utility) and eliminate any direct or quasi-identifier that could allow their re-identification (maximizing the user privacy). It should be as difficult as possible to relocate queries in order to build the original profile of a certain user.

3.3 Proposal

In Figure 3.1, three sub-systems that conform the proposed scheme are depicted. Note that, from those sub-systems, only the *WSE Anonymizer* should be integrated into the WSE environment. The other two are just defined to evaluate proposed method.

3.3.1 Actors

The main actors considered in the proposed process are the following:

- **WSE**: The web search engine. It builds the original query logs and it has the target of generating an anonymized version together with a users' profile database in order to sell or disclose them for economical purposes.
- **Customer**: Represents a third party who wants to legitimately access the anonymized query logs and the users' profile database.
- **Attacker**: Represents a third party who wants to gain illegitimate access to the original query logs using as input the anonymized query logs and the users' profile database.

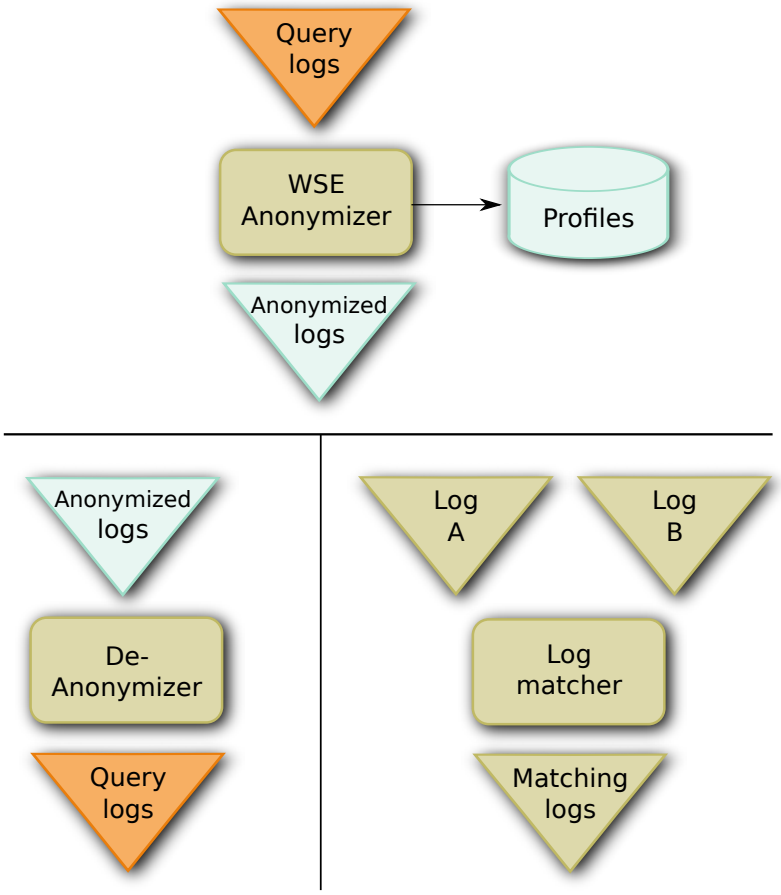

FIGURE 3.1: Main Architecture

3.3.2 Phases

Current study is divided into the following phases:

- **Anonymization and profile creation**: Main phase, which takes the original query logs generated by the WSE as input and generates the anonymized query logs and a database of user profiles as outputs.
- **Anonymization analysis**: Anonymization and performance benchmarking, taking into account original data, anonymization time and resource usage.
- **De-anonymization**: Using the anonymized query logs as input, an attack is simulated, trying to link anonymized logs with the users that issued them.

- **Analysis**: De-Anonymization and performance benchmarking, taking into account original and anonymized data, time and resource usage.

3.3.3 Interactions

As the system is designed to anonymize a stream of query logs in real time, interactions are defined by production and consumption of streams of logs. The main interaction is defined between a WSE as producer, and a customer as consumer of the anonymized logs. An attacker will try to gain access to the anonymized stream of logs, such as any legitimate client and, then, it will try to de-anonymize them in order to recover the original query logs.

3.3.4 Anonymization and profile creation

In Figure 3.2, the main modules of the anonymization and profile creation subsystems are represented. Each of them is responsible of a single action. All modules are described below.

3.3.4.1 Classifier

Represented in Algorithm 1, the classifier uses query logs generated on the WSE as main input. It also needs access to a database of entropies, recommendations and categories for a given word. Once the classifier receives an user query, it's processed and categorized in real time. Then the categorized query is released. The process followed by the classifier is divided into the following stages:

- **Natural language processing**: Which is applied to query's text field to extract its semantic units (SUs) [62]. Only nouns or adjectives are used. Then the HIPAA [146] Privacy Rule is applied, to remove all the SUs that could be considered identifiers or quasi-identifiers, such as a name, address, phone number etc.

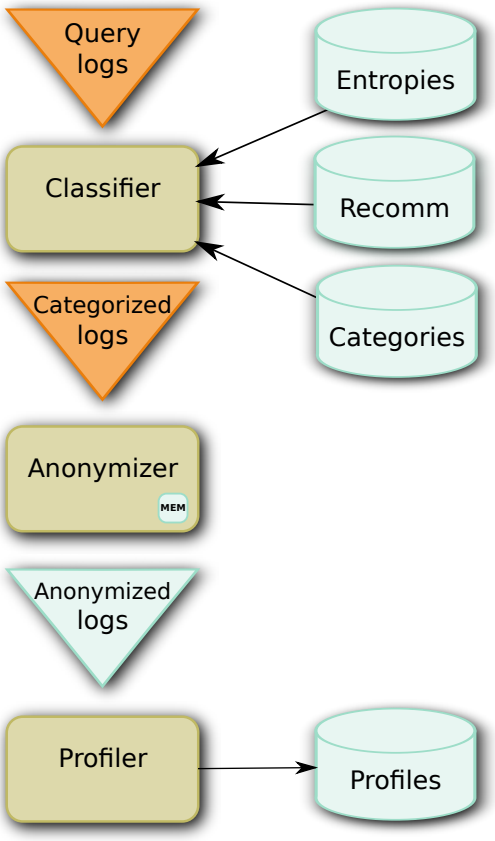

FIGURE 3.2: Anonymization and profile creation

- **Recommendation**: To correct possible misspellings, selected SUs are validated against a recommendation database, which returns the correct term in case of a misspelled word.

- **Entropies**: The amount of information (entropies) are calculated for each of the remaining SUs. To do so, a database of entropies is used, which should provide the number of references on the WSE for a given term. The algorithm will chose as *main_SU* the one which has less references, which is assumed to be the most specific and, hence, informative term of the query [147].

- **Categorization**: The corresponding category for the query is then calculated using *main_SU* and a database of categories, obtaining the most specific topic of the query.

As a representative example of how the *Classifier* works, let us assume an input query whose keywords are: "european soccer barcelona players". The first step

Algorithm 1: Classifier

Input : query_logs, entropies, recommendations, categories
Output: categorized_logs

```
1  foreach log ∈ query_logs do
2      SUs ← lemmatize(log);
3      min_entropy ← ∞;
4      foreach SU ∈ SUs do
5          l ← recommendation(SU);
6          e ← entropy(SU);
7          if e < min_entropy then
8              min_entropy ← e;
9              main_SU ← l;
10     end
11     log_category ← category(main_SU);
12     send log, log_category;
13 end
```

of this process extracts the SUs: "european soccer", "barcelona" and "players". Next, entropies are calculated for each SU, getting the following results according to Google's WSE: "european soccer": 56 500 000 results; "barcelona": 504 000 000 results; and "players": 544 000 000 results. According to these numbers, "european soccer" is selected as the *main_SU* of the input query. In the last step, "european soccer" (as *main_SU*) is used to assign a corresponding category of interest to the input query by means of a knowledge base. In this way, if the *Open Directory Project (ODP)* [1] is used as knowledge base and queried with "european soccer", retrieving "Sports" as the resulting category. Therefore, the input query is categorized as related to "Sports".

3.3.4.2 Anonymizer

Represented in Algorithm 2, it uses as input the categorized logs that are generated in real time by the classifier (see the previous explanation for more details about this). Those logs are split in two sets for each category, one that stores the users, and another that stores the text of the queries. When a category set reaches the maximum allowed value, defined by k, the algorithm randomly takes an user and

[1] Open Directory Project. `http://www.dmoz.org/`

a query from that category and builds with those the anonymized query with a minimum delay.

Algorithm 2: Anonymizer

Input : categorized_logs, k, δ
Output: Anonymized logs

```
1  foreach log ∈ categorized_logs do
2      user, query, category ← log;
3      users[category] ← user;
4      query[category] ← query;
5      if size(users[category]) = k then
6          if ∀u ∈ users[category], ∃u ≠ user then
7              pop random u ∈ users[category];
8              pop random q ∈ query[category];
9              send u, q;
10         else k = k * δ;
11 end
```

A special case occurs when all users stored in certain category set are the same. In this case, the selected user would be the same than the original one. In order to prevent this situation (which, in any case, it is very unlikely to happen due to the huge quantity of queries that a WSE receives each second), we impose an additional restriction, not shown in Algorithm 2 for simplicity: When all users in a certain category are the same, the anonymizer must increment the corresponding k value. This is done by multiplying k by a δ value.

As an example of how the proposed *Anonymizer* works, let us assume that the system works with parameter $k = 4$ and that in the data structure in charge of storing queries related to "Sports" there are already stored three queries: i) "novak djokovic tennis titles", sent by user A; ii) "monza formula1 tickets", sent by user B; and iii) "champions league final", sent by user C. The *Anonymizer* receives the query "european soccer barcelona players" that has been categorized as "Sports" in the last phase by the *Classifier*. The data structure "Sports" contains now four queries so, it is full, according to k; therefore, one of the stored queries is selected at random and it is assigned to a sender at random too. Following this example, let us assume that "champions league final" and user A are randomly selected. As a result, an anonymized log is outputted where user A is now linked to "champions

league final" instead of to her original query "novak djokovic tennis titles". It can be seen that, by doing this, the generality is kept (i.e., A is interested in "Sports") while the specificity is eliminated (i.e., A was specifically interested in "Tennis" instead of "Soccer"). As explained in works such as [148], keeping general interests improves the utility of the anonymized data while hiding specific interests is useful to preserve the privacy of the respondents.

3.3.4.3 Profiler

This element uses as input the anonymized record generated by the anonymizer and the corresponding category. With this data a user profile is created or updated in real-time. Therefore, the profiler is in charge of keeping the profile database updated. As it has been explained in the Introduction, in the literature, a user profile is generally considered a set of well-defined categories of interests (e.g., science, health, society, sports, etc.) with a certain weight assigned to each one according to the evidences generated by the corresponding user and how they have been classified under each category [11].

As an example of how the proposed *Profiler* works, let us assume that the system uses the following set of categories: Arts, Health, Shopping, Science, Computers, Sports, Society and Business (note that the proposed system is not bounded to this set of categories, this is only an example). Let us assume also that a certain individual has already sent: 5 queries related to Science; 10 queries related to Business; 20 queries related to Arts; 5 queries related to Health and 10 queries related to Computers. As a result, the current profile for that person stored in the profile database is: (Arts: 40%, Health: 10%, Shopping: 0%, Science: 10%, Computers: 20%, Sports: 0%, Society: 0%, Business: 20%). Now, let us assume that this person is assigned to the new query "champions league final" by the *Anonymizer*, this query has been categorized as "Sports" by the *Classifier*. The *Profiler* obtains the new evidence and updates the corresponding user profile as follows: (Arts: 39.2%, Health: 9.8%, Shopping: 0%, Science: 9.8%, Computers: 19.6%, Sports: 1.9%, Society: 0%, Business: 19.6%).

3.3.5 De-anonymization

The purpose of this process is to try to link the anonymized logs with the individuals who generated them. By this way, the system tries to evaluate whether the anonymization process executed previously has been successful or not and, therefore, whether the resulting protected query logs can be disclosed or not.

As seen in Figure 3.3, the de-anonymization process is very similar to the anonymization one. We assume that the attacker has already gained access to the stream of anonymized logs generated by the WSE. Those logs are the main input for the process.

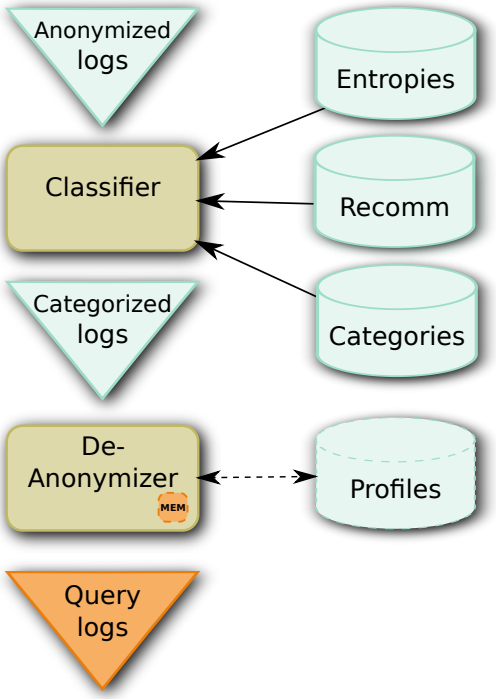

FIGURE 3.3: De-anonymization

First of all, the input logs should be classified by the attacker. Best de-anonymization could be achieved if the attacker manages to get the same categorization used by the WSE (this is also the worst-case scenario form a privacy point of view). As result, the attacker can use Algorithm 1 to categorize the anonymized logs prior to perform their de-anonymization. In this way, the attacker also needs access to a database of entropies, recommendations and categories.

Then, each categorized log is send to the de-anonymizer algorithm, which will try to recover the original query logs. Several de-anonymizer algorithms are proposed and tested on next section. All of them share the same basic structure, shown on Algorithm 3.

Algorithm 3: De-anonymizer

Input : categorized_logs, k, δ
Output: query_logs

```
1  foreach log ∈ categorized_logs do
2      user, query, category ← log;
3      users[category] ← user;
4      query[category] ← query;
5      if size(users[category]) = k then
6          if ∀u ∈ users[category], ∃u ≠ user then
7              pop_selected u ∈ users[category];
8              pop_selected q ∈ query[category];
9              send u, q;
10         else k = k * δ;
11 end
```

The de-anonymization algorithm also uses two additional parameters, k and δ. Their function is the same than in Algorithm 2. An attacker will always attempt to use similar values to the ones used in the anonymization process.

Each de-anonymizer variation defines a different *pop_selected* function, and some specific data structures, represented with dashed lines on Figure 3.3. All those variations are detailed in Section 3.4.

Following the same example used to explain the work of the *Anonymizer*, in this case, the *De-anonymizer* takes as input the anonymized query logs. Let us assume that this is: i) "champions league final", sent by A; ii) "monza formula1 tickets", sent by Z; iii) "novak djokovic tennis titles", sent by B; and iv) "european soccer barcelona players", sent by C. The *De-anonymizer* first categorizes the anonymized query logs to ascertain the category of interest linked to each query (as done in the anonymization process) and then tries to link a certain query with its original sender. The process followed to do that may vary and different approaches are explained in the Implementation section. A simple approach would be to match sender and query keywords at random (as done in the anonymization process).

Using this method in this example may lead to match B with "european soccer barcelona players" which would get a failure in the de-anonymization process, or with "monza formula1 tickets" which would get a success.

3.3.6 Analysis

Finally we also need a basic algorithm to verify that the proposed scheme is working properly, this is a log matcher. More concretely, it gets two log streams as input, and returns the number of matching logs, i.e. identical logs appearing in the two input streams. A resource profiler is also needed for a proper analysis, which at least should calculate amount of time and resources used in each task.

3.4 Evaluation

In this section, the implementation used to test our proposal is described. All the conducted tests are also detailed. Finally a discussion of system performance in terms of user's privacy, functional requirements and data utility is provided.

3.4.1 Implementation

All algorithms described in Section 3.2 were implemented in *Python*. Input and output query logs are stored in plain text files preserving the original format of logs. Communication between modules was also done via plain text files, to enable posterior analysis, but all modules could also use a sockets based communication. A *No-SQL* database, was used to store user profiles as well as other persistent data. Beside those implementation decisions, all other major changes that have been made to the initial algorithms during the implementation process, are discussed below.

Classification is a complex task to achieve outside of a WSE context, mainly because, besides the original logs, some additional information is required. As

previously described, the proposed algorithm needs a database of word entropies, a database of word recommendations and a database of word categories. In a WSE, this information would be provided by the WSE itself. It should be noted that a WSE already generates some of this information with each search, therefore it would not represent an extra cost. Outside of a WSE, this information should be retrieved querying an external WSE during the classification process but, by doing this, the obtained results in terms of time and resource usage may differ significantly from the ones that would be achieved in a real environment. In order to prevent this situation, a web scrapper was implemented as a preparatory step. The web scrapper retrieves information from remote sources and creates three local *No-SQL* databases that are the ones used in the classification step, therefore obtaining an environment similar to a real one.

Once classifying the data, we also need a way to lemmatize each query text, which is a phrase written on natural language. As the system was implemented in *Python*, the *NLTK* package (Natural Language Toolkit) [2]. This package provides interfaces to corpora and lexical resources such as *WordNet*, along with text processing libraries. On preliminary tests, *NLTK* was very slow for the requisites of our system, but due to the fact that it is a very generic system prepared for a wide variety of texts, uses and situations, a modified version of *NLTK* designed to fulfill our requirements was developed. This modified *NLTK* showed the same utility as the original package when dealing with query logs but it became hundreds orders of magnitude faster due to their focus on a more specific duty.

3.4.2 Evaluation methodology

In order to evaluate the architecture proposed in Section 3.2 we have implemented our system as described in Section 3.4.1. Only those systems which would be used on a real environment are evaluated regarding privacy, functional and utility requirements.

[2]Natural language toolkit. `http://www.nltk.org/`

TABLE 3.1: AOL query log example

1887264 5424618 ninja turtles rap lyrics 2006-05-22 17:04:54 1 http://www.lyrics007.com
1887267 5424618 myspace.com 2006-05-22 17:55:28 2 http://music.myspace.com
1887549 5426574 rio hotel and casino 2006-03-20 18:36:01 5 http://www.destination360.com
1887552 5426574 orleans hotel casino 2006-03-21 16:06:02 3 http://www.tickco.com
2536798 9146863 invest in spring drinking water 2006-03-07 13:51:21 2 http://www.fool.com
2536814 9146863 spring water stocks to buy 2006-03-13 22:13:44 4 http://importer.alibaba.com

3.4.2.1 Data

We ran our experiments on logs released by *AOL* [3] stored on plain text files, and on a database of word entropies, a database of word recommendations and a database of word categories, stored in a *NoSQL* database. Released *AOL* data contains 36 389 576 query logs, corresponding to a period of three months of real activity. Table 3.1 provides a brief sample of the used logs.

Databases of word entropies, recommendations and categories are created using the web scrapper defined in Subsection 3.4.1. Since the database content is generated based on logs content, databases only contain log related data, which can slightly affect system performance measures. A first attempt of data gathering was conducted using Google but it applies a very restrictive limit to the number of search queries from a certain source that can be served; as a result, our processes were blocked. Finally, *Microsoft Bing* was used to fill word entropies and recommendations databases. To create database of word categories, *Open Directory Project* (ODP) was used. ODP is a large, categorized directory of websites and pages, which is managed by volunteers.

Once the databases were created, no other query was done to any WSE, and all tests were conducted using those databases as the only data repository. At the end of scrapping process, databases contained 1 587 451 word categorizations, 1 751 632 word entropies and 258 504 word recommendations. Note that some query logs contain a query text with no understandable value. Grammatical mistakes were resolved using a word recommendation database. Other queries contain no query text, or it does not make any sense, being just lots of consonants concatenated. As a category for those last logs cannot be found, they were discarded.

[3] AOL keyword searches. `http://www.gregsadetsky.com/aol-data/`

3.4.2.2 Conducted tests

Only two parameters can be modified in the proposed system: k and δ. Therefore, several test were conducted to determine the effects of different k and δ values. It was also necessary some additional testing to determine the accomplishment for the rest of the requirements defined on Section 3.2.

δ adjustments — Some preliminary test were conducted to determine the effect of δ value. As explained on the algorithm definition, δ determines how fast k value increases when all k-elements on a category are the same. With a small δ the category size should be increased more times until we get different elements in the category. But with a small δ, the final k value fits best the needs of the data. With a large δ, category size should be increased less times, but the final k value should be bigger than the needs of the data.

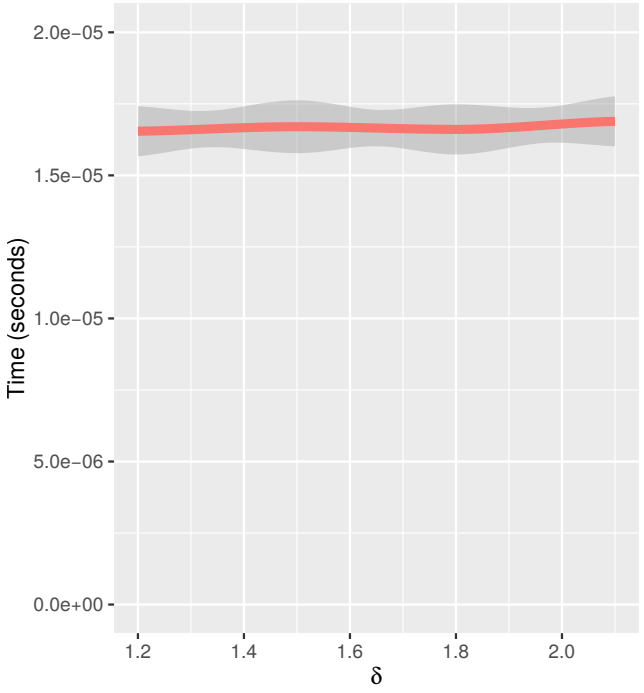

FIGURE 3.4: Effects of δ on time

Figure 3.4 shows that the size of δ does not affect the running time of the proposed system, hence, this is not a relevant factor to fix a certain δ value. Regarding the effect of δ on k value during the execution of the anonymization process, Figure 3.5 depicts that, even though bigger δ values produce slightly bigger final k

values reached by the system after multiple iterations, the difference obtained is not significant enough. Due to the fact that a small δ provides a more accurate final k value and smaller memory consumption without affecting other parameters, we decided to fix δ to 1.2 for the following tests. This leaves k as the only adjustable parameter of the proposal. As stated on following sections, the system performance, as well as the user privacy, depend on k.

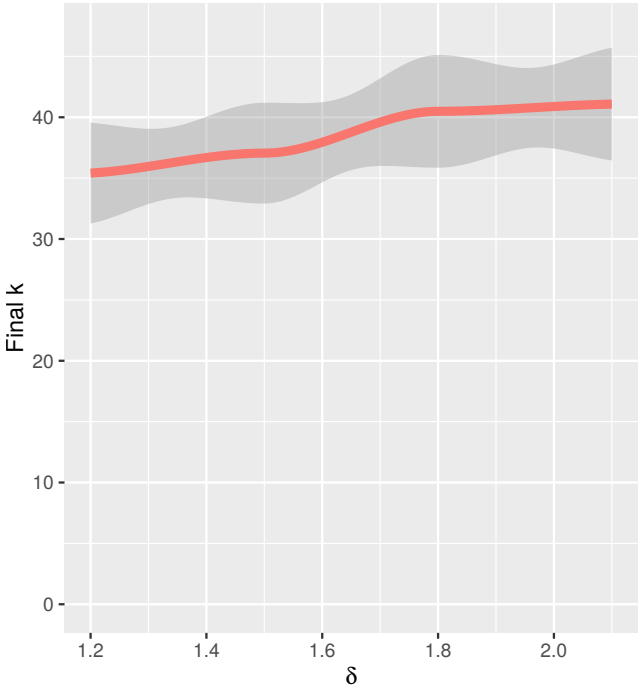

FIGURE 3.5: Effects of δ on final k

Classifier — The proposed classifier cannot be customized in any way; therefore, only some functional and utility tests were conducted.

Anonymizer — Privacy, functional and utility tests were conducted for the anonymizer, all of them with various values of k.

Profiler — The proposed profiler cannot be customized in any way; therefore, only some functional and utility tests were conducted.

De-anonymizer — In order to perform detailed privacy tests, some attacks were simulated using different variations of the de-anonymizer algorithm. In all these cases, the time field contained in the logs was used to consider the proper order in which each search query was received. We next detail each variation of the de-anonymizer element:

- *De-anonymizer 1*: This approach tries to retrieve original logs choosing one random log and one random user from the k-element sets, which the de-anonymizer recreates, conducting the same operations than the WSE.

- *De-anonymizer 2*: This approach does the same than the first version, but instead of selecting a random user, selects the user who appears more times in the k-element set linked to a certain category.

- *De-anonymizer 3*: This approach has access to the number of queries related to each category that were sent by this user until that moment. In this way, from the users who appear in the k-element set linked to a certain category on a given time, the proposed algorithm selects the user who has sent more queries related to this category. Therefore, this method makes its decision taking into account the query history of the respondents instead of just their temporal appearance in the k-element set linked to a certain category of interest.

- *De-anonymizer 4*: Uses the same approach than in the third version, but the number of queries related to each category that are obtained from the query history are multiplied by the number of times that the respondent appears in the k-element set linked to a certain category. The respondent with the highest result is then assigned to the current query. This approach considers the query history but tries to give more weight to the fact that an individual has sent a certain type of queries recently (which are those stored in the k-element set).

3.4.2.3 Test environment

All experiments were performed using a *Dell* notebook running *Ubuntu Linux* 14.04 LTS, with a 1.8 GHz *Intel Core*^{TM}i7-4500U CPU and 8GB of RAM. System hard disk was a *Seagate ST1000LM014*, which performance profile is skewed strongly towards small file I/O, and a below average overall performance. Much better results could be obtained with a faster hard disk or *Solid State Disks* (SSD), that are already installed in many servers nowadays. All algorithms were implemented and executed in *Python* 2.7.6. *MongoDB* 3.0.7 was used as *No-SQL* database, which was also installed and running on the same computer.

3.4.3 Privacy study

To test the privacy level provided by the new proposal, the original query logs were compared to the anonymized query logs, counting the percentage of matching records. Results can be seen on Figure 3.6.

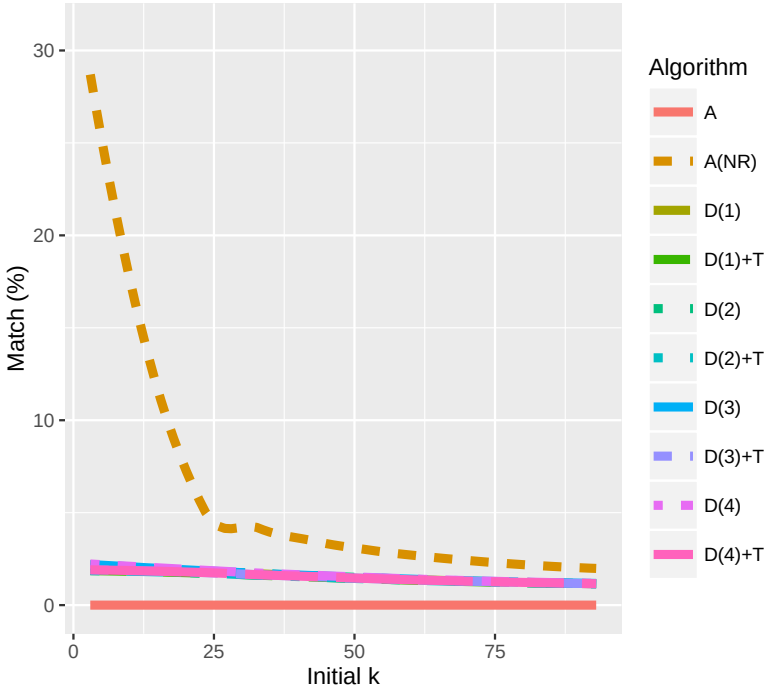

FIGURE 3.6: Matching records

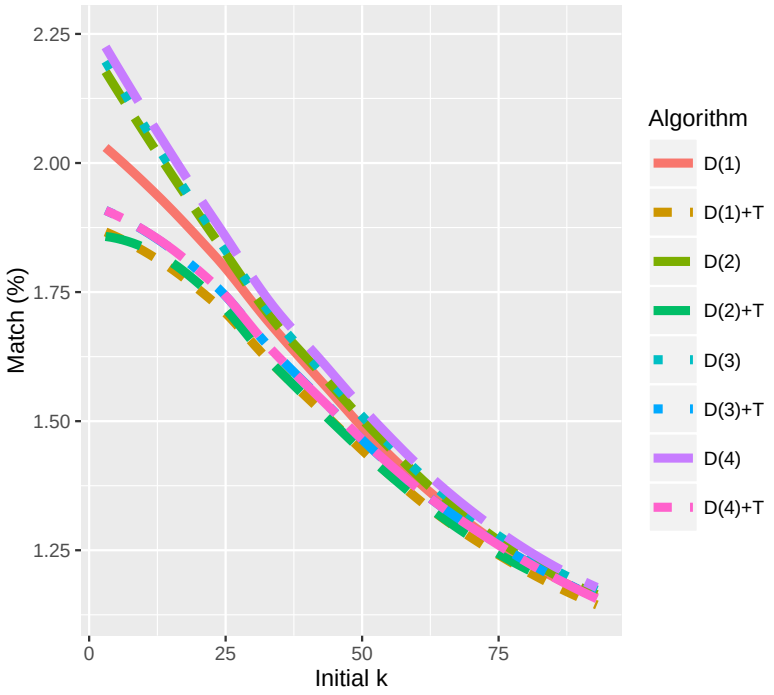

FIGURE 3.7: Matching records. De-anonymizer

In this figure, the case *A(NR)* represents what would happen if the anonymizer does not increment the corresponding k value when all users stored in certain category set are the same (this is a special case explained in Section 3.3.4). In this case, the figure depicts that k remains constant but the output contains non-anonymized logs. When this special case is covered by means of incrementing k, better privacy is obtained. Data generated by the anonymizer, A in the figure, do not contain any matching registers; therefore, full privacy seems to be obtained.

However, all de-anonymizer algorithms described above were used with the anonymized logs in order to try to reconstruct the original log stream. Results can be seen in Figure 3.6 and, more concretely, Figure 3.7 focuses on the four proposed de-anonymizers. The $+T$ element means that those de-anonymizers use logs sorted according to time field. In all the tests, all the de-anonymizer versions performed in a similar way (some differences applies but they are not significant) and they only were able to recover around 1.89% of original logs with the lowest k value (best case for the de-anonymizers). As a result of our tests, it can be concluded that it is quite difficult to link the anonymized logs with their original users.

3.4.4 Functional study

We next discuss the level of achievement of the functional requirements defined previously.

3.4.4.1 Modularity

The proposed system was developed as a set of independent services, forming a micro-service architecture. It allows to accomplish a low coupling and high cohesion system. This architecture makes the system more reliable and easy to maintain. Some services could be changed by existing modules in the WSE, e.g. a WSE classifier. Other services could optionally be deployed or not depending on the needs of the WSE. For example, the profiler should be used if the WSE is willing to create anonymous user profiles but, if the WSE only desires to release an anonymized log stream, the proposed system would also work without it.

3.4.4.2 Scalability

Thanks to the high modularity of the developed services, it is easy to deploy them on available idle systems, using the existing WSE infrastructure. Horizontal scalability could be achieved by deploying more than one instance of each service, either at the same or at different systems. Vertical scalability could be also achieved, since a faster CPU, disk or some memory dedicated to cache data would increase significantly the amount of work the system could handle. In lighter loads, modules will remain idle, consuming an insignificant amount of resources. Most modules could be closed completely if not used, for example, the classifier and the profiler. So this system shows a high load scalability.

3.4.4.3 Speed

Time consumption of WSE processes can be seen in Table 3.2. Even though the computer used for the tests has 4 cores, only 1 was used in each test. All algorithms support multi-threading, so better results could be obtained by simply enabling it.

Table 3.2: Runtime cost

	Time/log (μs) Mean ± SD	Queries/second (Threads Google)
Classifier	1 503 ± 24.0	665 (60)
Anonymizer	22 ± 0.35	45 454 (1)
Profiler	267 ± 4.73	3 745 (11)

The classifier and the profiler do not use k values and, hence, they are not affected by its variations. In the other hand, the anonymizer uses k, but results do not change significantly with different k values, as it is reflected by the small standard deviation.

The classifier was the slowest algorithm, because it has to search in several databases. We must be aware of two factors that affect its performance: i) the content of those databases was mainly generated using logs' content, therefore, databases only contain log related data; and ii) the majority of the time used by the classifier corresponds to I/O from disk, hence, a faster disk, or placing the data in the system's memory would greatly improve the performance of the classifier. The same idea could be applied to the profiler, which creates users profiles on a disk database.

Using as reference the 40 000 queries per second that Google processes on average, one single thread of the anonymizer in the used test environment can anonymize all Google queries in real-time. On the other hand, we need 60 classifier threads and 11 profiler threads to reach real-time performance in our test environment, but with the discussed changes, those numbers could be lower on a real setting.

For completeness, all de-anonymizer algorithms were also tested. Results can be seen in Figure 3.8. As expected, algorithms that count the number of times that

a user appears on a category, are the slowest overall, and more affected when k increases. Using profiles and choosing the best fit on a category was also affected by k.

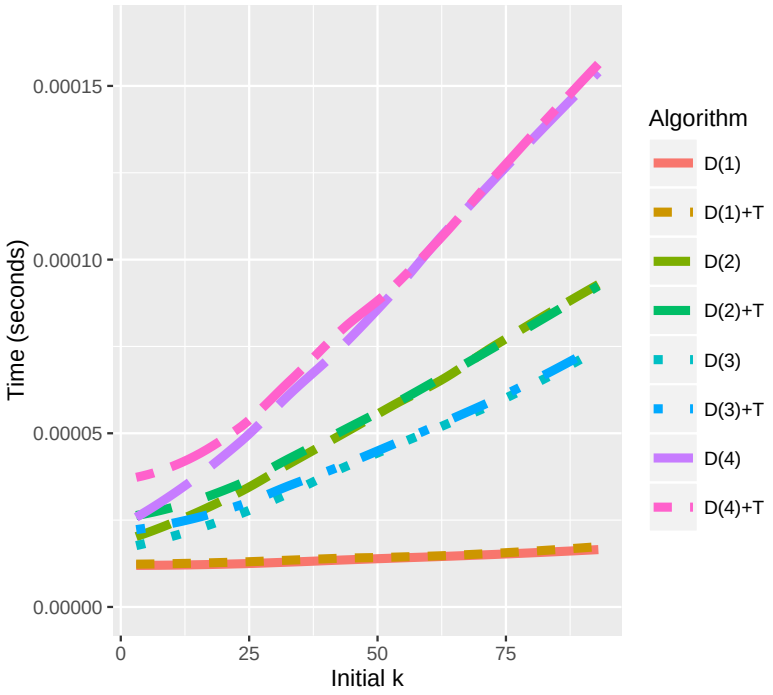

FIGURE 3.8: De-anonymizer speed

3.4.4.4 Resource consumption

Table 3.3 shows the achieved results for maximum memory and disk usage at each process. It's important to note that de-anonymizer-3 and de-anonymizer-4 keep basic user profiles in memory, and that is the reason because they consume more memory. Apart from those two algorithms, the rest should not use more memory, whatever the amount of logs they deal with.

Regarding the disk space, when output log streams are stored in the disk, they use exactly the same amount of space than the original log streams. Furthermore, if the profiler stores queries in a user profile, with 4GiB of test data, it generated a 3GiB profiles database, with an average record size of 112 bytes. Note that, in any case, this will depend on the input data.

TABLE 3.3: Memory usage

	Max. mem (KiB) Mean ± SD	
Classifier	111 581	± 145
Anonymizer	10 347	± 72
Profiler	12 120	± 94
De-anonymizer 1	9 316	± 640
De-anonymizer 2	9 375	± 687
De-anonymizer 3	312 273	± 4 620
De-anonymizer 4	311 782	± 2 945

3.4.4.5 Transparency

This proposal only needs to read the stream of logs already generated by a WSE, so it is no necessary to make changes at the existing WSE architecture. The resulting anonymized log stream has exactly the same structure and size than the original one. Therefore, this system can be plugged at the end of the current process followed by the WSE and generate an anonymized output without changing anything else.

3.4.5 Utility study

This section studies the classifier and the anonymizer in terms of utility. Even though the classifier obtains a hierarchical categorization for each query, the anonymizer sets are based only on very generic categories. When a query cannot be classified into any category, the classifier uses Microsoft Bing recommendations to find alternatives. As a result, the proposed classifier was capable of categorizing 85% of the all the tested queries. In order to verify that this automatic categorization was working properly, a sample of 1 068 logs were manually classified and compared to the results of the algorithm. This sample, in our population of 36 389 576 logs, provides a confidence level of 95%, with a margin of error of 3%. Through this comparison we found 58.99% of logs in a correct category. In Table 3.4 all used categories can be seen. Those categories correspond to the first

hierarchical level of ODP. Percentage of correctly classified logs for each category was also estimated.

TABLE 3.4: Classification utility

	Classification % Correct		Classification % Correct
Arts	57.23%	Recreation	55.22%
Business	77.39%	Reference	69.23%
Computers	64.79%	Regional	69.08%
Games	56.25%	Science	47.22%
Health	88.10%	Shopping	69.64%
Home	81.48%	Society	56.76%
Kids and Teens	37.65%	Sports	40.48%
News	16.67%	World	30.77%

It can be seen that this strategy works better in some categories. For instance, with categories that contain very specific terms, such as *Health*, the best results are obtained. In contrast, worse results are gathered in categories with generic terms, such as *News*. This is because the proposed algorithm only selects the word with lower entropy, which is expected to be the most specific one. Categorization was solely based on this word. When a combination of words changes the meaning of the query, this approach led to errors. For example, in the query "artic monkeys", the lowest entropy word is "artic". This word is categorized as "Regional: Polar Regions". That is correct for the word itself but using the whole context, category should be "Arts: Music". These results show that a better classifier should only be obtained by means of applying more complex natural language processing systems; however, this is outside of the scope of this proposal.

In addition to that, to check the anonymizer's utility, two sets of user profiles were created using the profiler. The first set contained profiles created with original logs while the second set contained profiles created with anonymized logs. Both sets were compared in order to check whether they were equivalent. It was confirmed that, for a given user, both sets of profiles contained the same number of queries for each category. We argue that if an anonymized profile contains the same categories with the same weights than an original profile, it implies that the anonymized profile retains the same utility than the original one.

3.5 Conclusions

This chapter has presented a novel framework for anonymizing query logs generated by WSEs. Privacy guarantee is defined in terms of set theory, which relates sets of users to sets of query logs. Data could be released without other modifications than removing direct identifiers from query text and remapping between those two sets. This contrasts to existing approaches that release heavily modified data, either distorted or generalized, to maintain anonymity.

In our evaluation, we have considered the worst-case scenario, in which an attacker who is willing to use the anonymized query logs to retrieve the original query logs has gained access to the same base information and algorithms than the WSE. We conducted tests under this context and the best attempt to recover the original logs only obtained a 1.89% of them. All the proposed methods were tested using the AOL released logs; therefore, we argue that our solution is capable of dealing with real data in a real setup. In this way, we have considered also the average Google's load to study the runtime cost and the memory usage. The query log's utility after its anonymization was also analyzed, and the results showed that original query logs and anonymized query logs were equivalent in terms of categories being reflected.

Chapter 4

A Real-Time Query Log Protection Method for Web Search Engines

4.1 Introduction

As previously discussed, query logs are fundamental for the proper operation of several services offered over the Internet, but also can lead to some severe problems, related to users' privacy. In the last chapter, we obtained query logs with high degree of privacy using a *statistical de-identification* approach. Our proposal was able to process a stream of query logs in real time and generate anonymized logs with similar queries from disctinct users.

In this chapter, we want to improve the previously proposed system. We assume WSEs are seeking to monetize query logs by making them available to third parties, while respecting privacy regulations [38]. In order to make data monetization viable, one also must keep in mind that reducing privacy disclosure risk is not enough. At the same time, as much data utility as possible should be preserved. In the previous chapter, query logs were classified using broad categories, which caused a certain degree of utility loss.

We reason some modifications over the previous approach to achieve a system with customizable data utility, in addition to the proposed use of probabilistic k-anonymity to bound disclosure risk of personally identifiable user attributes. Therefore, the present chapter expands our previous proposal to use parameterizable levels of categorization. We seek to enable the WSE to adjust utility of generated logs according to the needs of each application. We will also validate the proposal to prove that it is able to achieve the same or better level of privacy than previously reported, with an improved overall data utility. Formal and experimental proofs will be devised to ensure its feasibility.

The rest of this chapter is organised as follows: Section 4.2 presents our proposal. Section 4.3 provides architectural components and requirements. Section 4.4 provides experimentation results validating our approach. Section 4.5 concludes the chapter.

4.2 Our Proposal

We present in this section our anonymization proposal. Table 4.1 introduces the notation used along this section. Next, we provide a formal definition of the expected data we aim to anonymize, the way how the data is structured, a formal analysis about the privacy properties of the proposal, and the algorithmic version of our anonymization process.

4.2.1 Data Structures

We assume a stream of query logs, formed by m registers, where r_m corresponds to the last received query log:

$$R = \{r_0, .., r_m\} \tag{4.1}$$

TABLE 4.1: Notations used in this chapter.

R	:	Stream of query logs
r_j	:	Individual query log
u_i	:	User unique identifier
q_j	:	Individual query text
c_q	:	Full individual query classification
τ	:	Hierarchical ontology of categories
V	:	Set of vertices
v_x^h	:	Vertex at depth h depth and width x
E	:	Set of edges
e_f	:	Edge between two vertices
Q_x^h	:	Set of queries for vertex v_x^h
U_x^h	:	Set of users for vertex v_x^h
γ_x^h	:	Category for vertex v_x^h
$\tau_{\ell,k}^*$	:	τ with a depth ℓ and width k

Each register is of the form:

$$r_j = \{u_i, q_j, c_g\} \tag{4.2}$$

where u_i is a unique identifier that represents the user who sent the query q_j to the WSE. Each query q_j is composed of a set of unstructured terms, which we previously provided to a categorizer (cf. Section 4.3) to obtain the classification of the query, denoted as c_g. This classification c_g is represented as the path from a general category γ_s^1 to a more specific category $\gamma_{s^*}^h$, with the form:

$$c_g = \{\gamma_s^1, \gamma_{s'}^2, ..., \gamma_{s^*}^h\} \tag{4.3}$$

The path is created according to a hierarchical ontology structure by means of a tree structure τ, which is formed by a set of edges $e_f \in E$ and vertices $v_x^h \in V$, where h is the depth and x the width. Each vertex v_x^h of τ represents a category γ_x^h, and is related to other categories through the edges. The vertices or categories are more generic the closer they are to the roots $\{v_1^1 ... v_x^1\}$, and more specific the closer to the leaves. Thus, every query is classified by assigning it to one of the vertices of the tree. As mentioned, the classification is the path between the root and the vertex, and it is composed by all the γ categories of the nodes that are in the path.

$$\begin{aligned} \tau &=< V, E > \\ V &= \{v_1^1, ..., v_z^\ell\} \\ E &= \{e_1, ..., e_g\} \\ v_x^h &= \{U_x^h, Q_x^h, \gamma_x^h\} \\ e_f &= \{v_x^h, v_{x'}^{h+1}\} \end{aligned} \tag{4.4}$$

The maximum depth of the hierarchy τ is ℓ_{max}, defined as the distance or minimum path between the root and its farthest leaf. The number of terms or depth for each classification may be ℓ_{max} or lower, but we will use limited versions at depths up to ℓ, where ℓ goes from 1 to ℓ_{max}.

Each vertex v_x^h contains a set of users U_x^h, and a set of queries Q_n^m. The size of U_n^m will be k, but the size of Q_x^h may be larger. This is because U is defined using arity, but Q is defined without the need of using arity.

$$max \mid U_x^h \mid = k \tag{4.5}$$

$$max \mid Q_x^h \mid \geq k \tag{4.6}$$

Therefore, we call $\tau_{\ell,k}^*$ the tree τ with a depth ℓ and a value of $|U| = k$.

4.2.2 Restrictions

To properly explain why U_x^h and Q_x^h may have different size, we introduce two additional restrictions that we impose to our proposal (cf. Restrictions 4.1 to 4.2).

Restriction 4.1. A given query associated to an anonymized log must not be assigned to the same user that issued the query on the unanonymized log.

Restriction 4.2. When creating an anonymized query log, user must be selected randomly between at least k different user values.

Restriction 4.1 ensures that outputs do not contain unanonymized pairs of user and query. Restriction 4.2 imposes probabilistic k-anonymity, setting at least k distinct values for users in each category when randomly creating an anonymized log.

4.2.3 Anonymization Process

We define our anonymization process as the method that generates the probabilistic k-anonymous stream of logs R':

$$R' = \{r'_0, ..., r'_m\} \tag{4.7}$$

We assume that each record $r_j = \{u_i, q_j, c_g\}$ in R is assigned to the corresponding v_x^h using its categorization c_g. The record r_j is then separated in two parts: u_i which is assigned to U_x^h, and q_j which is assigned to Q_x^h. Records in R' are obtained by applying a random match between one element of U_x^h and one element of Q_x^h, once $\mid U_x^h \mid = k$:

$$r'_j = \{u'_i, q_j, c_g\} \tag{4.8}$$

where $q_j \in Q_x^h$ is matched with a $u'_i \in U_x^h \neq u_i$.

The Id function is assumed to be a correct identification function, which given r'_j responds with the original u_i. The function Re is a re-identification function used over the records in R', which given a r'_j responds with:

$$Re(r'_j) = u_i \in U_x^h, u_j \neq u'_i \tag{4.9}$$

The goal of probabilistic k-anonymity is to limit the probability of performing the right re-identification to at most $\frac{1}{k}$ for all $u_i \in R$ and for all the values of $Re(r'_j)$:

$$P(Re(r'_j) = Id(r'_j)) \leq \frac{1}{k} \tag{4.10}$$

The stream of logs R' is said to satisfy probabilistic k-anonymity if, by knowing R' and the anonymization process, the probability to link any record $r'_j \in R'$ and its corresponding record $r_j \in R$ is, at most, $\frac{1}{k}$.

We show next that our proposal satisfies the property defined in Eq. (4.10). For each vertex v_x^h of τ, the random selection of an element (Restriction 4.2) guarantees that all outcomes are equally likely to be selected. Therefore, we can state maximum probability of re-identification of a r'_j over τ using:

$$P(Re(r'_j) = Id(r'_j)) \leq \max_{\forall x,h} \frac{|U_x^h \cap Id(r'_j)|}{|U_x^h|} \tag{4.11}$$

As U_x^h sets are defined using arity, we know that:

$$\forall x, h, Id(r'_j) \rightarrow |U_x^h \cap Id(r'_j)| \in 0, 1 \tag{4.12}$$

Someone could argue that Restriction 4.1 leads to a value of $k - 1$. However, since Restriction 4.2 establishes this value to k (Restriction 4.2 also assures that $|U_x^h| \geq k$), the upper bound of our proposal for $P(Re(r'_j) = Id(r'_j))$ is strictly lower or equal to $\frac{1}{k}$, hence satisfying probabilistic k-anonymity. A more formal analysis about this result is provided next.

4.2.4 Privacy Analysis

Given k (anonymity parameter) in $\mathbb{Z}^+$, a set of users $\mathcal{U}$ equal to $u_1, ..., u_n$ (such that $n \geq k$), a set of query logs $\mathcal{Q}$ equal to $(u_{i_j}, q_j)_{j=1}^{j}$ up to the processing iteration

j, where $q_k \neq q_l \ \forall k, l \in [j], (k \neq l)$, $u_{i_j} \in \mathcal{U}$. We also assume that users repeat (i.e. $u_{i_k} = u_{i_l}$).

We assume that given a query in R', the whole R' and k, an arbitrary PPT (Probabilistic Polynomial-Time) adversary $\mathcal{A}$ has at most $\frac{1}{k}$ chance of guessing the user the given query was attached to in R.

Now, with the notation above, and let $j_0 \in [j]$ define and experiment $Exp_{Re}(k, R)$, in which:

$$\begin{aligned} R' &\leftarrow Anon(k, R) \\ R^* &\leftarrow Re(k, R') \\ let\ b &= \begin{cases} 1, & \text{if } R = R^* \\ 0, & \text{otherwise} \end{cases} \\ & return\ b \end{aligned} \tag{4.13}$$

Theorem 1. Anon (cf. Eq.(4.13)) is probabilistic k-anonymous if, for every user set, for every query log R and every index $j_0 \in [j]$, any PPT adversary $\mathcal{A}$ has a bounded advantage up to $\frac{1}{k}$, i.e.,

$$\begin{aligned} Adv_{\mathcal{A}}(k, R) = \\ P[Exp_{Re}(k, R)] \leq \frac{1}{k} \end{aligned} \tag{4.14}$$

Proof. Let $R' = (u'_{i_j}, q'_j)_{j=1}^{j'}$ and j the iteration at which the first log entry is released by the anonymizer after (u, q) has been read by itself. Let $\mathcal{U}_j^{R'} = (u_{i_{j_1}}, ..., u_{i_J})$ be the users presents at R' at iteration j and $\mathcal{U}_j = (u_{i_1}, ..., u_{i_k})$ be the user set used internally in the anonymizer at iteration j (i.e., we know $u \in \mathcal{U}_j \in \mathcal{U}_j^{R'}$ and $\mathcal{U}_j$ has at least k different users).

$$\begin{aligned} P(\mathcal{A}(R', q) = u) = \\ \sum_{u' \in \mathcal{U}} P(\mathcal{A}(R', q) = \ u | (u', q) \in R) \cdot P((u', q) \in R) \end{aligned}$$

If U_j and Q_j are the users and queries stored by the anonymizer after reading query q, where U_j has at least k different users, permute users from the queries of Q_j to R (all in U_j) has no effect on the anonymizer output, i.e.:

$$P(\mathcal{A}(R', q) = u | R = Re(R')) = \sum_{u \in \mathcal{U}_j} P(\mathcal{A}(R', q) = u | [R = Re(R')] \cap [U_j = U]) \cdot P(U_j = U)$$

where U_j contains the users that can appear in step j, hence $u \in \mathcal{U}$. If U_j is fixed and $u \in U_j$, we can consider an R where the query q is paired with each of the users u' of U_j, and one of the queries q' whence the entries of u' from U_j are now paired with U.

If we have read j_u times the user u, $\forall i : j_i \geq 1$, we obtain that the ratio of R^*s, being $R^* = Re(R')$ and $U_j = U$, which contain the original pair (u, q) is:

$$P(\mathcal{A}(R', q) = u | R = Re(R')) = \frac{(j_{u_2} + ... + j_{u_k} + j_{u-1})!}{(j_{u_2} + ... + j_{u_k} + j_u)!} = \frac{1}{(j_{u_2} + ... + j_{u_k} + j_u)} \leq \frac{1}{k}$$

hence satisfying Theorem 1. □

4.2.5 Algorithmic Version of our Proposal

An algorithmic version of our anonymization process is presented in Algorithm 4. Algorithm 5 presents the anonymization process counterpart, assumed to be implemented by a PPT adversary. Algorithm 4 receives three main inputs: desired k, ℓ values, and R as a stream of hierarchically categorized query logs.

Even if all the sets are initialized empty, our proposed algorithm guarantees that U_x^h is of size k every time a new anonymized log is generated from that category. It also tries to keep the Q_x^h size as close as possible to the k value. As it always

chooses between k different users and at least k different queries, probabilistic k-anonymity is guaranteed.

Q_x^h size may be bigger than k in the following situation: each time a new log enters a category and the log's user was already present on that category, user's arity is increased by one in U_x^h and the query is added to Q_x^h. Therefore, $|U_x^h|$ stays the same but $|Q_x^h|$ is increased by one. If Restriction 4.2 is not met, there is no anonymized log release (i.e., the size of Q_x^h can be bigger than k).

If Restriction 4.2 is met, and some user's arity is greater than one, then Algorithm 4 releases an additional log to reduce the size of Q and user arity, also enforcing Restriction 4.1. This extra step is only done once per log, therefore at most two logs are generated each time a new record enters the category, until all users' arities are equal to one.

Algorithm 4: Anonymization Process

Input : R, k, ℓ
Output: R'

foreach $r_j \in R$ **do**
  // Get current user, query text and full query categorization
  $u, q, c \leftarrow r_j$;
  // Truncate categorization to level ℓ
  $cat \leftarrow \{\gamma_s^1, ..., \gamma_{s^*}^\ell\} \in c$;
  // Add current user to users' category set
  $users[cat] \leftarrow u$;
  // Add current query and full categorization to queries' category set
  $query[cat] \leftarrow \{q, c\}$;
  // While there are more than k distinct users on the current category
  while $distinct(users[cat]) > k$ **do**
    // Select and remove a random query and categorization from the category's set
    pop random $\{q', c'\} \in query[cat]$;
    // Select and remove a random user from the category's set, distinct from the original user related to the query
    pop random $u' \in users[cat], u' \neq Id(q)$;
    // Send to the output the selected user, query and category
    send u', q', c';
  end
end

Algorithm 5: De-Anonymization Process

Input : R', k, ℓ
Output: R*

foreach $r'_j \in R'$ **do**
  // Get current user, query text and full query categorization
  $u, q, c \leftarrow r'_j$;
  // Truncate categorization to level ℓ
  $cat \leftarrow \{\gamma_s^1, ..., \gamma_{s^*}^\ell\} \in c$;
  // Add current user to users' category set
  $users[cat] \leftarrow u$;
  // Add current query and full categorization to queries' category set
  $query[cat] \leftarrow \{q, c\}$;
  // While there are more than k distinct users on the current category
  while $distinct(users[cat]) > k$ **do**
    // Select and remove a query and categorization from the category's set, using one of the record linkage algorithms
    record_linkage $\{q', c'\} \in query[cat]$;
    // Select and remove a user from the category's set, using one of the record linkage algorithms
    record_linkage $u \in users[cat]$;
    // Send to the output the selected user, query and category
    send u, q, c;
  end
end

System performance remains stable whenever variations of the set size is proportionally conducted (cf. Chapter 3). Hence, we modify the size of each set in incremental unitary steps. This allows the most efficient memory usage. In addition to the k parameter, the depth of categories' tree must be specified using the ℓ parameter. Both k and ℓ remain fixed to the specified value throughout the entire execution.

Tables 4.2 and 4.3 depicts an example using $k = 2$ and $\ell = 1$. These values have been chosen to facilitate the understanding of the example, but they are inferior to desirable values in a real application of the algorithm (cf. Section 4.4). The example starts with an empty system, receiving a stream R of query logs classified in two distinct categories. Figure 4.1 depicts the used R, and the contents of τ and R' at the end of the aforementioned example. Figure 4.2 depicts the deanonymization counterpart, leading to faulty re-identification.

STEP 1: first query arrives

INPUT: r_1\{u=Alice, q="piano", c=Arts/Music\}

users[Arts] = Alice
query[Arts] = "piano"

STEP 2: second query goes to a new category

INPUT: r_2\{u=Bob, q="myspace", c=Computers/Internet\}

users[Arts] = Alice
query[Arts] = "piano"
users[Computers] = Bob
query[Computers] = "myspace"

STEP 3, 4: still no distinct users $> k$ in any category

INPUT: r_3\{u=Alice, q="guitar", c=Arts/Music\}
INPUT: r_4\{u=Charlie, q="violin", c=Arts/Music\}

users[Arts] = Alice, Alice, Charlie
query[Arts] = "piano", "guitar", "violin"
users[Computers] = Bob
query[Computers] = "myspace"

STEP 5: distinct users $> k$ in "Arts", but k after the first output

INPUT: r_5\{u=Bob, q="flute", c=Arts/Music\}

users[Arts] = Alice, Alice, Charlie, Bob
query[Arts] = "piano", "guitar", "violin", "flute"
users[Computers] = Bob
query[Computers] = "myspace"

Category full: distinct(users[Arts]) = $3 > k$
OUTPUT: r'_1\{u=Charlie, q="piano", c=Arts/Music\}

users[Arts] = Alice, Alice, Bob
query[Arts] = "guitar", "violin", "flute"
users[Computers] = Bob
query[Computers] = "myspace"

TABLE 4.2: Applying Algorithm 4 with k=2 and ℓ=1 (Part I)

STEP 6: new query, but no disctinct users $> k$ in any category

INPUT: r_6\{u=Charlie, q="google", c=Computers/Internet\}

users[Arts] = Alice, Alice, Bob
query[Arts] = "guitar", "violin", "flute"
users[Computers] = Bob, Charlie
query[Computers] = "myspace", "google"

STEP 7: distinct users $> k$ in "Computers"

INPUT: r_7\{u=Alice, q="aol", c=Computers/Internet\}

users[Arts] = Alice, Alice, Bob
query[Arts] = "guitar", "violin", "flute"
users[Computers] = Bob, Charlie, Alice
query[Computers] = "myspace", "google", "aol"

Category full: distinct(users[Computers]) = $3 > k$
OUTPUT: r'_2\{u=Bob, q="google", c=Computers/Internet\}

users[Arts] = Alice, Alice, Bob
query[Arts] = "guitar", "violin", "flute"
users[Computers] = Charlie, Alice
query[Computers] = "myspace", "aol"

STEP 8: distinct users $> k$ in "Arts"; k after the second output

INPUT: r_8\{u=Charlie, q="drums", c=Arts/Music\}

users[Arts] = Alice, Alice, Bob, Charlie
query[Arts] = "guitar", "violin", "flute", "drums"
users[Computers] = Charlie, Alice
query[Computers] = "myspace", "aol"

Category full: distinct(users[Arts]) = $3 > k$
OUTPUT: r'_3\{u=Alice, q="drums", c=Arts/Music\}
Category full: distinct(users[Arts]) = $3 > k$
OUTPUT: r'_4\{u=Bob, q="guitar", c=Arts/Music\}

users[Arts] = Alice, Charlie
query[Arts] = "violin", "flute"
users[Computers] = Charlie, Alice
query[Computers] = "myspace", "aol"

TABLE 4.3: Applying Algorithm 4 with k=2 and ℓ=1 (Part II)

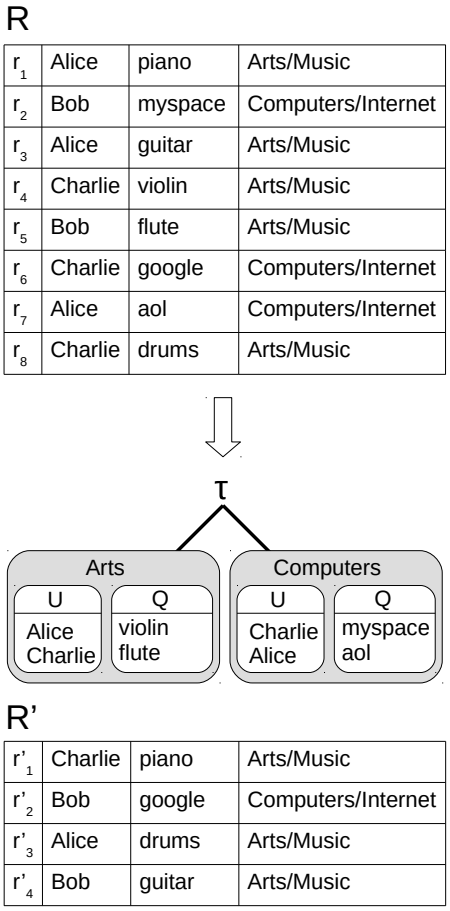

FIGURE 4.1: Contents of R, τ and R' in the example provided in Tables 4.2,4.3.

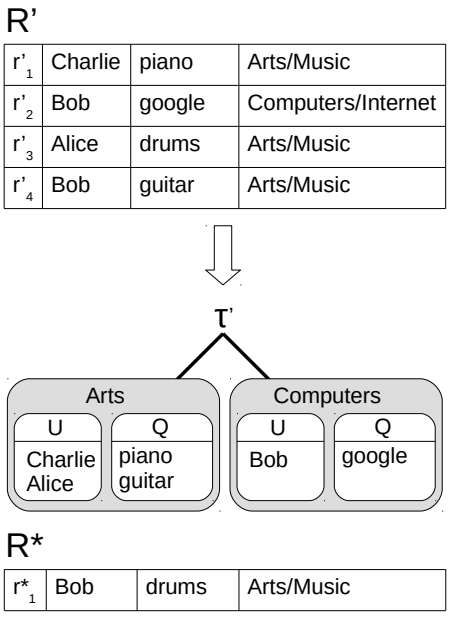

FIGURE 4.2: Contents of τ' and R^* when trying to deanonymize R' from the example provided in Tables 4.2,4.3

4.3 Practical Implementation

We present in this section a practical implementation of our proposal. We describe the architecture and requirements, before moving to the presentation of the experimental results.

4.3.1 Initial Architecture

We aim at implementing an anonymization method that can be used by Web Search Engines (WSEs) to anonymize query logs in a streaming environment, and at server-side (cf. Figure 4.3). The input data of the anonymization algorithm is a continuous stream of categorized query logs. The outputs are a continuous stream of anonymized logs and a database of user profiles. To meet the goals of our proposal, we must ensure that those outputs meet a set of requirements detailed below.

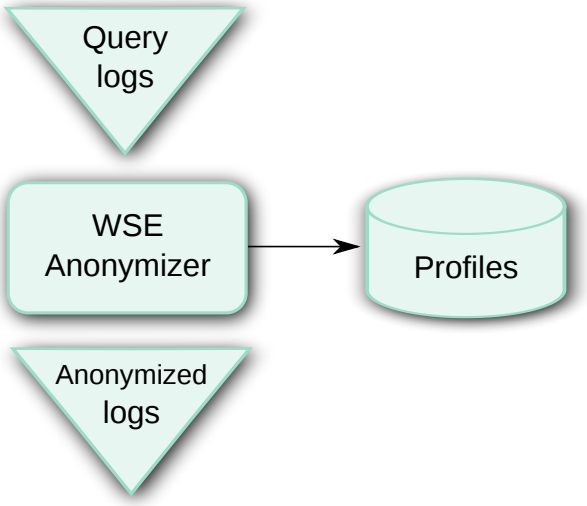

FIGURE 4.3: Our proposal defines a WSE query logs anonymization method in a streaming environment. The input of the algorithm is a stream of query logs. The outputs are a stream of anonymized logs and a database of user profiles.

4.3.2 Functional Requirements

In addition to the restrictions and properties already defined in Section 4.2, we report next some functional requirements for the practical implementation of our proposal.

Scalability — It refers to the capability of a system to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth [149]. In our system, the objective is to achieve load scalability, defined as the ability to accommodate heavier or lighter loads. Those methods can be classified in two main categories [150]:

- **Horizontal Scalability** is related to the ability of a system to add more working nodes, such as a new computer. Hundreds of small computers may be configured in a cluster to obtain aggregate computing power. This approach demands an architecture that allows efficient management and maintenance of multiple nodes.
- **Vertical Scalability** is related to the ability of adding resources to a single node in a system, typically involving the addition of CPUs or memory. Such approach could be interesting in a virtualized environment, as it could provide more resources according to the virtual node needs. This approach demands an architecture that allows efficient management of used processes and memory.

The two models have their own particular benefits and limitations. If necessary, our proposal should use all possible assets. In such a case, the design should be integrated into existing systems on a WSE architecture. Ideally, our system can take advantage of underused resources.

Resource Consumption — In order to take advantage of underused resources on existing architectures, and minimize system deployment costs, we want a minimal resource consumption. If the designed system is able to use a limited amount of

resources, all necessary data could be kept and processed in memory, obtaining better execution times.

Speed — We need a fast processing speed to be able to process all received logs in real time. Otherwise, some kind of memory buffer will be necessary to keep incoming logs until processed. That buffer will increment our resource consumption. An additional requirement, in terms of processing speed, must be defined and only use small buffers at specific overload times. Nowadays, a WSE receives millions of user queries each hour. Therefore, our system should handle that load, to be able to integrate it in a existing WSE architecture.

Efficiency — Beyond reduced resource consumption and fast processing time, we aim at assuring the algorithmic efficiency of the proposal. We consider that this requirement will be achieved if the algorithmic time complexity of our proposal is linear according to the inputs.

Transparency — We want a straightforward integration of our approach into an existing architecture. Having a transparent system implies that no component of the existing WSE should be modified. For this purpose, our module is expected to be encapsulated within the WSE. It should also be able to interact to the existing interfaces of the WSE, without forcing any changes. It should also be able to generate anonymized logs, while complying with all the previous requirements.

Modularity — We want to have low coupling and high cohesion to achieve a fully transparent component. Modularity has the added benefit that modifications to the proposal could be implemented with minimal effort, as well as to carry out tests with different alternatives for the treatment of the data.

4.3.3 Expanded Architecture

The initial proposal depicted in Figure 4.3 is expanded with two additional parts: Attacker and Researcher. This allows a proper empirical evaluation, in addition

to the analysis conducted in Section 4.2. The proposed system is designed using a micro-service architecture pattern as presented in Figure 4.4. For the current study, all the defined systems are used. In a real WSE environment, only the parts marked as WSE should be deployed.

Within the expanded architecture, we find two main components: anonymizer and profiler. The anonymizer is a component implementing Algorithm 4. The profiler creates protected user profiles, using the categories of each log assigned to that user by the *anonymizer*. Those categories are added to a user profile database in real-time. Each profile on the database contains a frequency distribution of those categories queried by the user. They can be seen as user interests that could be released to third parties, for profit.

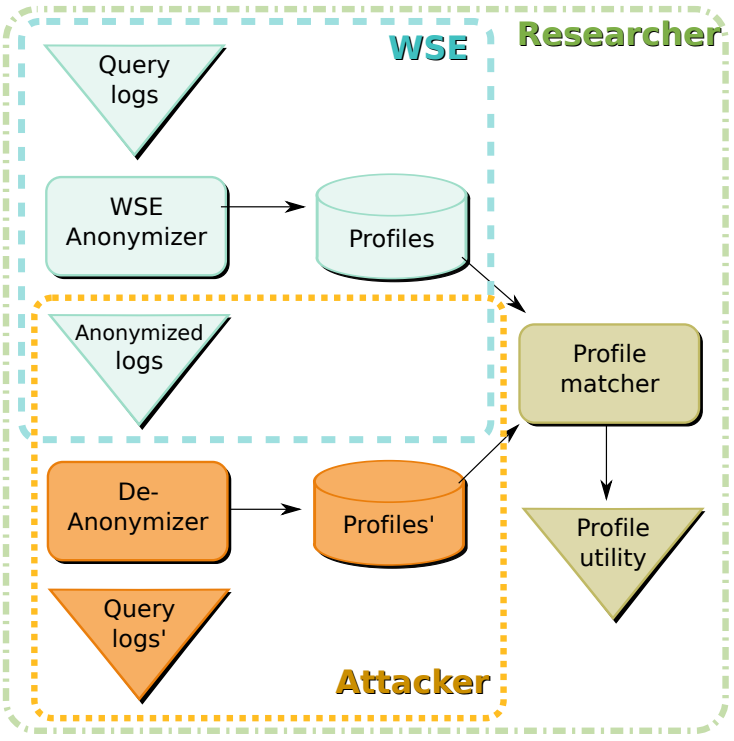

FIGURE 4.4: Full Architecture: *WSE Anonymizer* takes a stream of query logs and anonymizes them, also generating a database of user profiles. It implements Algorithm 4 (cf. Section 4.2). *De-anonymizer* implements Algorithm 5 and simulates adversarial actions over the anonymized logs. It tries to recreate the original logs and user profiles. *Profile matcher*, responsible of benchmarking anonymization, de-anonymization and performance, also generates a profile utility metric.

4.3.3.1 Actors

Three actors are defined in our current test architecture:

- **WSE** — has the responsibility of query logs anonymization and publication.
- **Attacker** — has access to the anonymized stream of logs, tries to recover the original relationship between the log and the user who made the original query.
- **Researcher** — can check all the data, but can not modify anything, to test the validity of the proposal.

4.3.3.2 Phases

Our study is divided into three main phases:

- **Anonymization and profile creation** — this phase represents the normal execution of the system on the WSE environment. It takes the query logs generated and anonymizes them, also generating a database of user profiles.
- **De-anonymization** — it simulates attacks, trying to link as much of the anonymized logs with the user that originally made the query.
- **Analysis** — it conducts anonymization, de-anonymization and performance benchmarking, taking into account original and generated data, time and resource usage.

4.3.3.3 Interactions

In a real WSE environment, the WSE will anonymize the query logs and release the anonymized ones to its clients as the main interaction. In our tests, the attacker is acting as a normal client from the WSE point of view. The attacker process the anonymized output of the WSE and generates another log stream, trying to

reconstruct original query logs. Only during the tests, secondary interactions occur between those actors and the researcher, who receives original, anonymized and de-anonymized query logs. Some further information about it is presented in the sequel.

4.4 Experimental Results

We report in this section a practical implementation of our approach, and report experimental tests and results, to validate our approach in terms of privacy, data utility and other functional requirements.

Experiments were conducted using a *Dell* notebook running *Ubuntu Linux* 16.04 LTS, with a 1.8 GHz *Intel Core*^{TM}i7-4500U CPU and 8GB of RAM. System hard disk was a *Seagate ST1000LM014*, whose performance profile is skewed strongly towards small file I/O, and a below average overall performance. All algorithms were implemented and executed in *Python* 2.7.12.

4.4.1 Implementation

Algorithm 4, described in Section 4.2, has been implemented using the *Python* language. Input query logs used to test our system were downloaded from the public available AOL log repository, in form of plain text files. In order to respect our transparency functional requirement, we chose to make this file the main input of our system. However, other methods to feed logs to the system, such as a real time input via sockets, could be used. The same applies to system output and we also decided to store them in plain text files, preserving original logs' format. Additionally, a *No-SQL* database was used to store generated user profiles.

Because AOL's released files do not have any classification, they need to be categorized by an external categorizer before any of the proposed algorithms could be applied. We used a modified version of the deterministic classifier proposed in Chapter 3. The use of a deterministic classifier guarantees that the same query

will always provide the same unique category. In case a query triggers multiple categories, the classifier will always take the most probable one. Other families of classifiers can be adapted and integrated in our approach thanks to the proposed micro-service architecture. Classifier modifications allow us to obtain a query categorization organized in several hierarchical levels. Some queries contain letters or symbols without any meaning, and some contain no text at all. Our classifier was not able to resolve those logs, and they were left out of data used to test the proposal. However, some changes made to natural language processing algorithms on the *classifier* lead to categorize 98% of original logs, an improvement of over the 85% categorized in Chapter 3. As it is out of the scope of the current proposal, implementation of the *classifier* will not be evaluated. Priority will be given to allow interoperability between our proposal and different *classifiers*. Usually, classification process needs more specific data, related to WSE environment or desired output categories. Thus, we leave freedom to each WSE to choose the strategy that best suits their needs.

We also validate the possible record linkage of the anonymized stream, implementing three different record linkage algorithms, and evaluate for each algorithm whose requirements are fulfilled. In addition, some other changes that have been made to the initial architecture described in Section 4.3 are discussed below.

4.4.2 Evaluation Methodology

The algorithmic solution proposed in Section 4.2, and all the architectural components, requirements and implementation details defined in Sections 4.3 and 4.4.1, have been used to conduct an experimental evaluation and comparison to previous work in Chapter 3. In particular, one version of the anonymizer, and three versions of the de-anonymizer are implemented and evaluated in terms of utility, privacy and functional requirements.

4.4.2.1 Experimental Datasets

For our experiments, we use plain datasets (i.e., text files), containing query logs released by *AOL* [151]. The released *AOL* data contains up to thirty six million (36 389 576) query logs. Such query logs correspond to a three-month period of real web search activity conducted by *AOL* users, and released by *AOL* for research purposes. Figure 4.5 provides a brief sample of the used logs.

```
116874 thompson water seal   2006-05-24 11:31:36 1 www.thompsonswaterseal.com
116874 express-scripts       2006-05-30 07:56:03 1 www.express-scripts.com
116874 express-scripts       2006-05-30 07:56:03 2 member.express-scripts.com
116874 knbt                  2006-05-31 07:57:28
116874 knbt.com              2006-05-31 08:09:30 1 www.knbt.com
117020 naughty thoughts      2006-03-01 08:33:07 2 www.naughtythoughts.com
117020 really eighteen       2006-03-01 15:49:55 2 www.reallyeighteen.com
117020 texas penal code      2006-03-03 17:57:38 1 www.capitol.state.tx.us
117020 hooks texas           2006-03-08 09:47:08
117020 homicide hooks texas  2006-03-08 09:47:35
117020 homicide bowie county 2006-03-08 09:48:25 6 www.tdcj.state.tx.us
117020 texarkana gazette     2006-03-08 09:50:20 1 www.texarkanagazette.com
117020 tdcj                  2006-03-08 09:52:36 1 www.tdcj.state.tx.us
117020 naughty thoughts      2006-03-11 00:04:40 1 www.naughtythoughts.com
117020 cupid.com             2006-03-11 00:08:50
```

FIGURE 4.5: AOL log format. Each row represents a query log. Columns contain, from left to right: user identifier, query submitted, time submitted, result selected and result URL.

The Classifier (cf. Section 4.4.1), adds to each log record an additional column with a hierarchical classification in form of a list with n elements. In our case, n was between one and 13, and each element of the list represents a subcategory of the previous element. This classification is generated independently of the anonymizer. Therefore, this list contains all the subcategories which the Classifier is able to generate for a given query, regardless of the ℓ used by the anonymization process.

4.4.2.2 Conducted Tests

Proposed system could be configured using two parameters: k and ℓ, being k the desired number of different users on each category and ℓ the maximum depth of categories and subcategories used for each record. Several tests were conducted to determine its effects.

Anonymizer To generate anonymized data, proposed *anonymizer* was executed on all available AOL logs multiple times, to cover different k and ℓ values. k has taken values between 3 and 200 to be able to compare obtained results with previous ones in Chapter 3. To do this, Algorithm 4 needs to be tested at least using $\ell = 1$. We decided to test all available ℓ values, that with our classification correspond to values between one and 13, but we found that from 11 onwards, differences were not significant: few logs have more than 11 categories of depth. Our privacy, functional and utility requirements are checked for every combination of k and ℓ.

Profiler Specific tests were conducted with the *profiler*, to determine the amount of data utility that could be lost with anonymized profiles creation respect to unanonymized profiles. For those tests, we used k values between three and 90 and ℓ values between one and 13.

De-anonymizer A de-anonymization has been attempted against all anonymized data. All anonymized data was tested against three different *record-linkage* algorithms:

- **Record-linkage 1** — This is the simplest record-linkage algorithm we tested. It tries to apply an inverse transformation to anonymized query logs by applying a similar algorithm to the one used in the anonymization process (cf. Algorithm 4 in Section 4.2). In short, it tries to recreate original logs by randomly matching users and queries from the same category. Attacker also takes advantage of both restrictions 4.1, 4.2 to achieve higher levels of de-anonymization.
- **Record-linkage 2** — It improves the performance over *Record-linkage 1*. Instead of randomly matching users and queries, it assigns the user that appears more times on a category to the selected query. Just like other algorithms, both restrictions are respected.

- **Record-linkage 3** — It keeps track of how many times a user issued a query on each category, constantly updating a simplified user profile. When the algorithm needs to assign a user to a query, the user with more issued queries on that category will be chosen. If a user appears more than one time, the result will be multiplied by the number of appearances of that user, balancing the importance between current state of the system and historical values.

4.4.3 Privacy Study

Our privacy test compares original query logs data with the anonymized ones. Results for this base case show that none of the original pairs of user/query appear on the anonymized query log. Notwithstanding, that result did not guarantee full user privacy, since some attacks are possible over the output data flow, and some user logs may be re-identified. Three different record-linkage algorithms were applied to the anonymized query logs (cf. Algorithm 5). Resulting logs were compared to the original ones, counting the percentage of matching records.

Our de-anonymization algorithms proposal is based on Algorithm 5, that is similar to Algorithm 4 used in anonymization. It uses the stream of anonymized logs generated by the WSE as the main input. It also needs k and ℓ parameters (explained in Section 4.2). The smaller the difference between k and ℓ values used in both algorithms, the better the results obtained from de-anonymization. In other words, the attacker will be able to re-identify the original data more easily.

The stream of anonymized logs is classified in the same way as the original one, since we assume that categorization is public and the attacker can use it. Therefore, the de-anonymization process uses the same categorization, which enables this algorithm to obtain the best de-anonymization rate when trying to recover the original logs.

The main difference with the *anonymizer* algorithm is the use of *record_linkage* function, different for each implementation of the *de-anonymizer* algorithms. The most complex de-anonymizers also use additional data structures to improve de-anonymization performance. Differences of each algorithm are fully explained in Section 4.4.2.2.

For analysis purposes, we need to evaluate the amount of memory and time used in each algorithm execution, therefore, previous algorithms were modified to calculate those values. An additional algorithm must be defined to find the number of logs that are identical comparing two log streams.

Figure 4.6 shows percentage of matching records, executing the three algorithms with values of k between three and 200 and values of ℓ between one and 13. With $\ell = 1$, only one level of the tree structure was used, which results in a data structure equivalent to the one used in Chapter 3. $\ell = 13$ is the maximum depth that our classifier was able to generate. Thus, there is no need to use higher ℓ values. We also picked out k values to be able to compare results between our current and former evaluation.

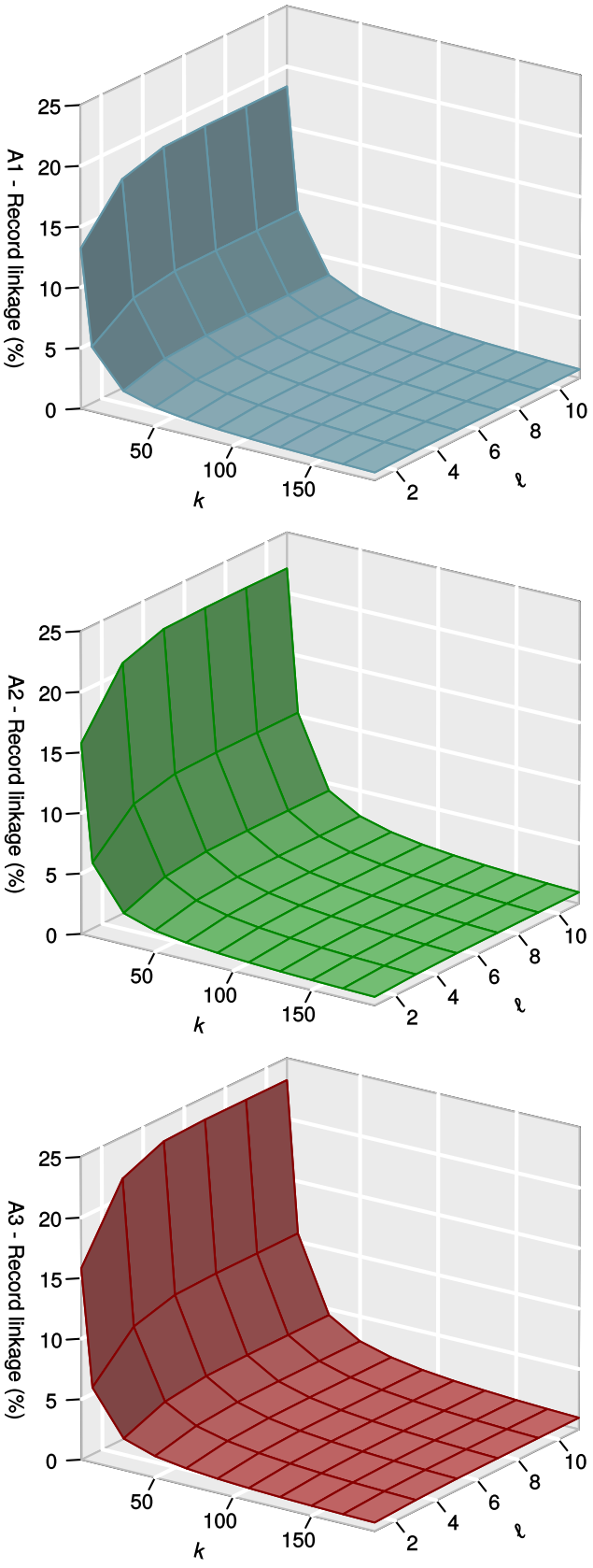

FIGURE 4.6: Record linkage (%). Percentage of matching records, executing the three de-anonymization algorithms with values of k between three and 200 and values of ℓ between one and 13.

In all cases, results are under the theoretical maximum probability $\frac{1}{k}$ of being re-identified [152]. We ran the Kolmogorov-Smirnov goodness-of-fit statistical test [153, 154] to compare the k-anonymity probability with the experimental results, Figure 4.7. The maximum difference between the cumulative distributions, D, is 0.08 with a corresponding p-value of 0.9977. Therefore, the statistical test yields to acceptance of the null hypothesis that our results follow k-anonymity's probability of re-identification (at the 5% level of significance).

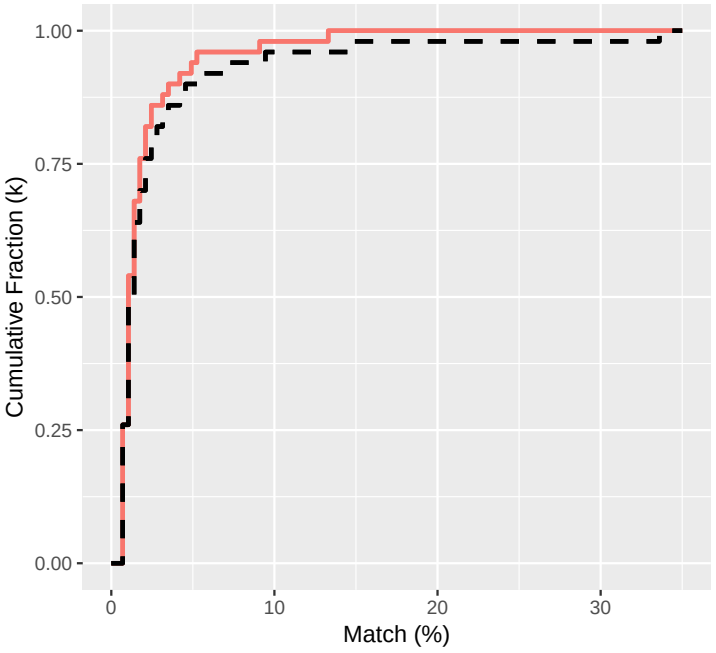

FIGURE 4.7: Comparison between cumulative fraction functions of the theoretical k-anonymity (dashed) and experimental results (solid). Used for the Kolmogorov-Smirnov test ($D = 0.08$, p-value$= 0.9977$).

Each *record-linkage* version improves re-identification rate, being the third version the one that obtains better results overall. k value was highly correlated with privacy, because when the value of k increases, record linkage decreases. ℓ also affects privacy. With a higher number of levels (high ℓ value) users were matched with more specific queries, therefore, it was also more probable to obtain a correct re-identification of the original user. Here, we face a trade-off between privacy and data utility.

Results obtained this way, are close to the ones obtained in our previous article using the proposed algorithm without restriction, since now the effective size of

the category sets are closer to the k value specified as a parameter. However, on average a better anonymization is obtained, since the size of Q must be temporarily increased to meet the restrictions 4.1 and 4.2.

Figure 4.8 shows mean final $|Q|$ values, related to ℓ and initial k value. For low ℓ values, mean final $|Q|$ values are higher because they have less categories results and more user coincidences on the same category. However, with small k and ℓ values, the high number of queries that passes through each category counter this effect. With higher ℓ values, final $|Q|$ values tend to match up with specified k.

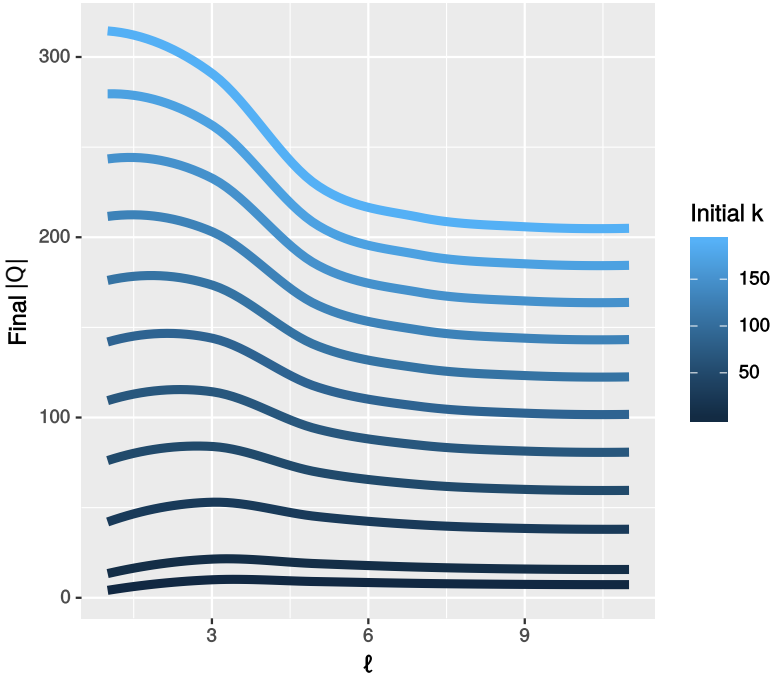

FIGURE 4.8: Final $|Q|$-value. For low ℓ values, final $|Q|$ is higher due to more user coincidences on the same category. With higher ℓ values, final $|Q|$ tend to match the specified k.

The highest record linkage is obtained with highest ℓ and lowest k values. Our best de-anonymizer algorithm was able to link 23.18% records to the original user. De-anonymization tests were conducted knowing exactly all algorithms, categories and variables used for anonymization. This ratio decreases quickly when initial k value is increased, obtaining a record linkage lower than 1% from k values greater than 90. In conclusion, desired record linkage level could be adjusted by modifying the k value, even offsetting the effect of ℓ variations on the record linkage.

4.4.4 Utility Study

We proceed to analyze the utility of the proposed anonymizer. This analysis has been focused on two different aspects:

- Percentage of logs that the system can generate as an output.
- Preservation of original user's interest in anonymized user's profiles.

First, we want to analyze the percentage of logs that can be generated by the system over the total number of logs that it gets. The proposed system uses sets, and each set must have at least k different users before being able to release an anonymized log. A possible drawback to this approach is that some sets do not reach k users and, therefore, the logs contained in this set do not end up leaving the system. As we can see in Figure 4.9, this effect exists and it is directly proportional to the depth of the category tree. This is consistent, since with more depth, more categories are created and the minimum of k users on these categories is reached more slowly. However, we see that as more queries enter the system, all categories become filled with queries and the percentage of log output increases, tending to a 100% rate for any depth of the tree.

Secondly, to measure the preservation of original user's interest in anonymized user's profiles, we will measure the distance between them, using a metric known as Earth Mover's Distance (EMD) [155].

We calculate the distance between the categories of queries assigned to the original profile and the anonymized profile. As our classification of categories is stored in a tree graph, this distance is defined as the minimum length of the path that connects the categories assigned to the original and anonymized query. Once we have calculated the distance between individual queries, we add all the distances of that profile and, thus, we obtain the total distance between profiles.

Notice that if two queries are classified and anonymized with the same category, there is no distance between the two queries and there is no utility loss. This

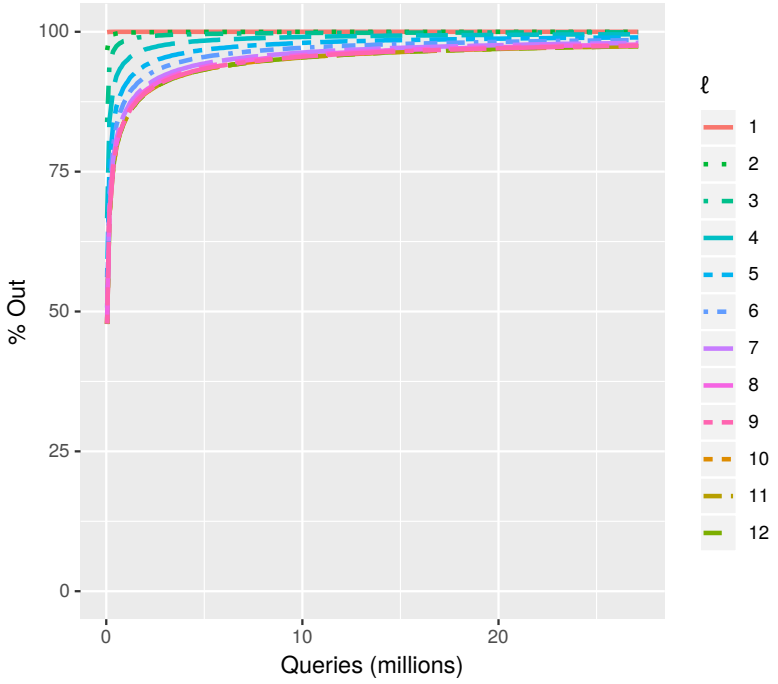

FIGURE 4.9: Output queries vs. total queries (%). Some sets take a while to fill. This effect is directly proportional to the depth of the category tree as more sets need to get k different users.

happens to all the queries when the depth of the tree is set to 13. However, other tree depths can lead to utility loss. For instance, in the example of Tables 4.2, 4.3, "piano" is classified as "Arts/Music" but the anonymizer is just using "Arts", since the value of ℓ is equal to 1. Queries classified as "Arts/Music" and "Arts/Painting" are mixed in "Arts" and assigned to different users. A third party could think that Alice is interested in "Painting", when she is just interested in "Music". i.e., there is a certain degree of utility loss. Since the third party still knows that Alice is interested in "Art", we can see the previous case as an example of partial utility loss. Therefore EMD represents the distance between the original user's interests, and the ones that are deducted from the anonymized queries.

In Figure 4.10, we can see the average value of the EMD distances, as well as the maximum theoretical distance between profiles using the chosen categorization. This theoretical maximum distance is constant, regardless of which ℓ and k values we use. The real distance we get is not affected by k, but is inversely proportional to ℓ. This means that the more levels we use in our anonymizer, the closer the anonymized queries get to their original category and we obtain a better data

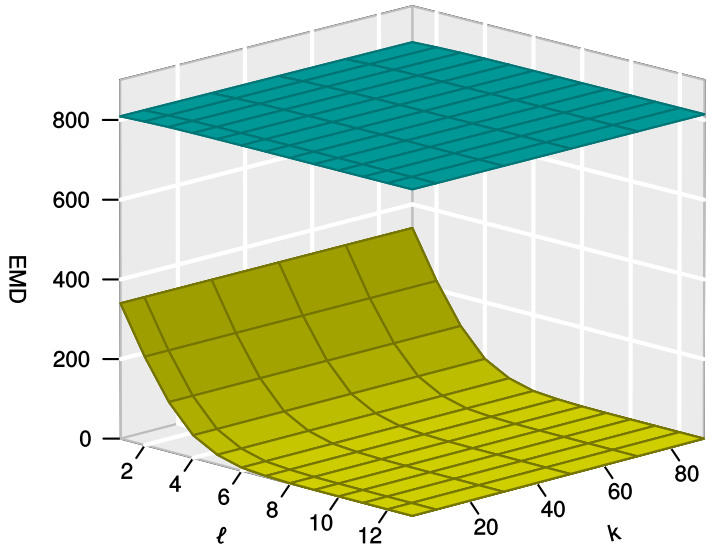

FIGURE 4.10: Maximum theoretical distance between profiles, constant, and average EMD distances, inversely proportional to ℓ.

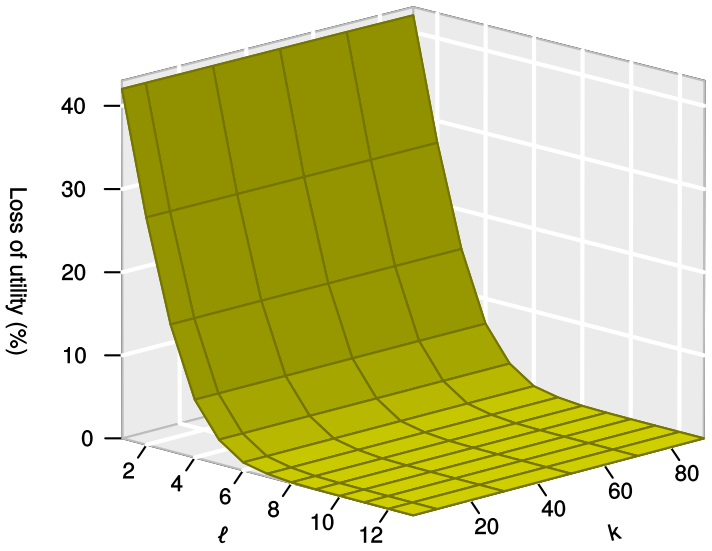

FIGURE 4.11: Loss of utility (%), the more levels we use in our anonymizer, the better data utility.

utility. In Figure 4.11, we can see the loss of utility expressed as a percentage. Using this metric, it can be seen that with $\ell = 1$, loss of utility is over 40% on average. With $\ell = 6$, the loss of utility is near to 0%, according to our definition of utility.

4.4.5 Functional Study

Next, we detail the accomplishment of proposed functional requirements.

4.4.5.1 Modularity

To allow a modular system, this has been designed as a set of micro-services. As our proposal uses micro-service architecture, it will be easier to modify and adapt when applied to different environments. In addition, this design helps each service to focus only on a specific process. By doing so, we achieve a system with low coupling and high cohesion. The anonymization service has been thoroughly explained. This service can be connected to other modules such as categorization and profile creation.

4.4.5.2 Scalability

The proposed system can be scaled, both vertically and horizontally. Vertical scalability is achieved by varying the number of resources assigned to the system. These resources can be added either in form of memory or CPU cycles. Horizontal scalability can also be achieved by activating or deactivating different instances in parallel. In addition, with the proposed anonymizer, the value of k could be dynamically adjusted, which also allows to improve the scalability of the system using it in a wider range of situations.

4.4.5.3 Speed

Speed of the anonymizer and deanonymizers was tested. All the results that are shown correspond to the time required to completely treat a query using a single thread of execution on a single core. All the proposed algorithms can be used in parallel, achieving a better system throughput.

The fastest execution was achieved with $k = 3$ and $\ell = 1$, where on average a query was processed in 18.99 μs. Therefore, the system can handle up to 52659 queries per second, on average.

Average processing time per query was 33.68 μs, or 29691 queries per second. It includes executions with all the k and ℓ values we have tested. Compared to our previous proposal where we obtained 22 μs per query, we see that the system is slower on average, but with greater data utility. However, depending on which parameter values are used, the system is faster than our previous proposal, as described below.

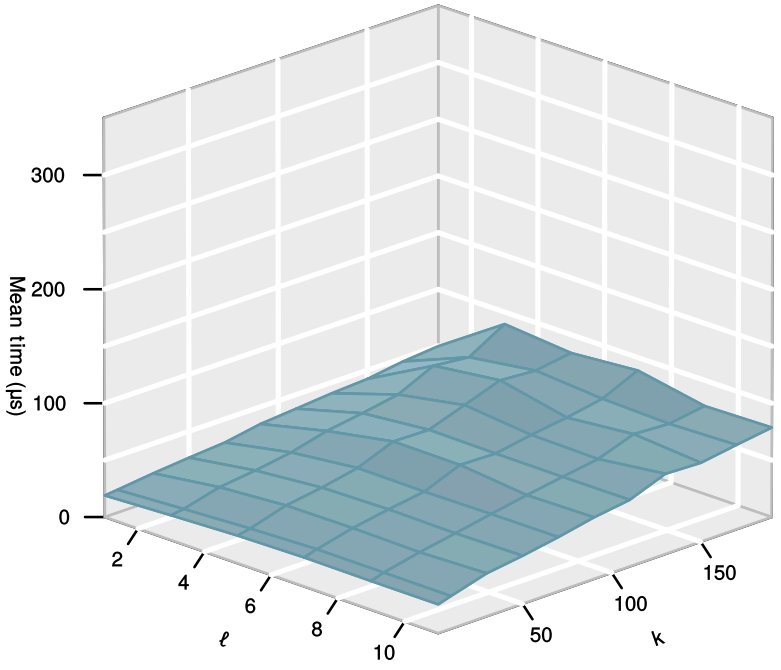

FIGURE 4.12: Anonymizer mean time per query (μs). ℓ-value has little effect on required time, k-value has a greater effect.

Speed of the anonymizer is affected by k and ℓ. If we look at Figure 4.12, we can see that changes in the value of ℓ have little effect on required time. Contrarily, changes in the value of k have an important effect. For example, for $k = 3$ the system can process a log in about 18.99 μs. This value reaches 49.71 μs with a a value of $k = 190$. Taking into account that Google treats an average of 40000 queries per second (cf. Ref. [156] and citations thereof), a thread of our algorithm

could handle all real-time queries, using k-values up to 50 with any value of ℓ, according to our test results.

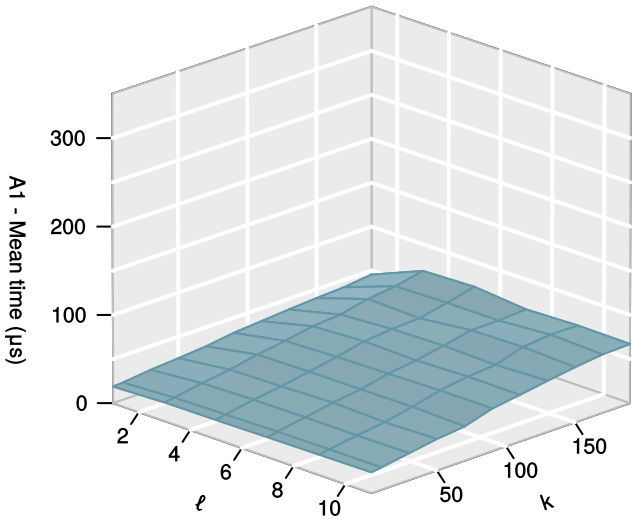

FIGURE 4.13: De-anonymizer 1 - Mean time per query (μs). First de-anonymizer obtained comparable results to the anonymizer.

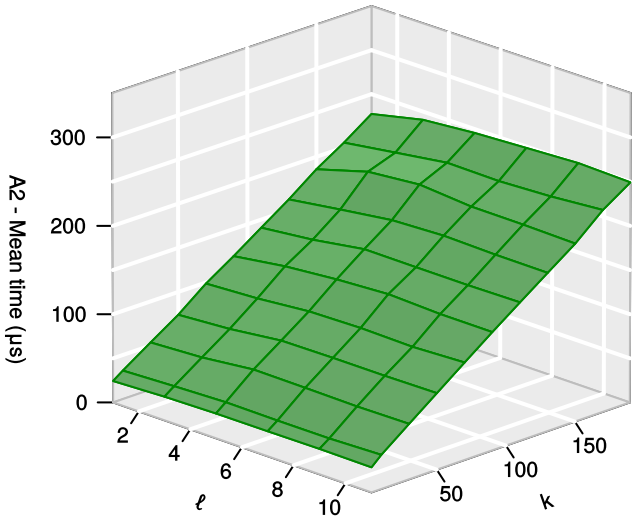

FIGURE 4.14: De-anonymizer 2 - Mean time per query (μs). Second de-anonymizer, which is more complex, is also slower and more affected by increases in k-value.

The same analysis has also been done with proposed de-anonymization algorithms. Results can be seen in Figures 4.13 4.14 4.15. The first de-anonymizer approach obtained results comparable to the anonymizer. This was expected since in both cases the same base algorithm was used. Second and third de-anonymizers, which

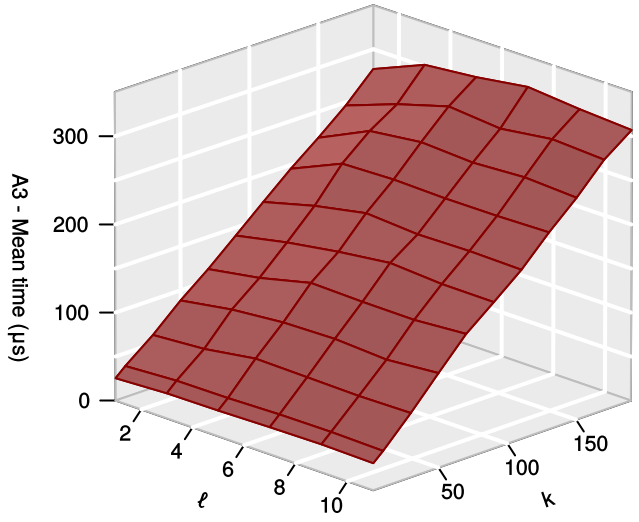

FIGURE 4.15: De-anonymizer 3 - Mean time per query (μs). Third de-anonymizer, the most complex, is also the slowest, being more noticeable with high k-values.

perform more complex operations, are also slower and more affected by increases in k-values. In all cases, we see that variations of ℓ-values are less important.

4.4.5.4 Delay

Another factor that we consider important to evaluate is the average delay of queries between entering and leaving the system in form of anonymized query logs. Figure 4.16 shows this delay as the mean number of other queries that enter the system during the period between the entry and the release of a given query. As we can see, this delay is increased proportionally to the chosen ℓ-value, but it ends up stabilizing. This is reasonable, since the system needs to fill categories initially and once this happens, the output stabilizes.

Taking as reference the 40000 queries per second that Google receives (according to Ref. [156]), we see that our system's output stabilizes in a few minutes for larger values of ℓ. Once the delay is stable, our system takes less than one second for values $\ell \leq 6$, and does not reach two seconds for larger values of ℓ.

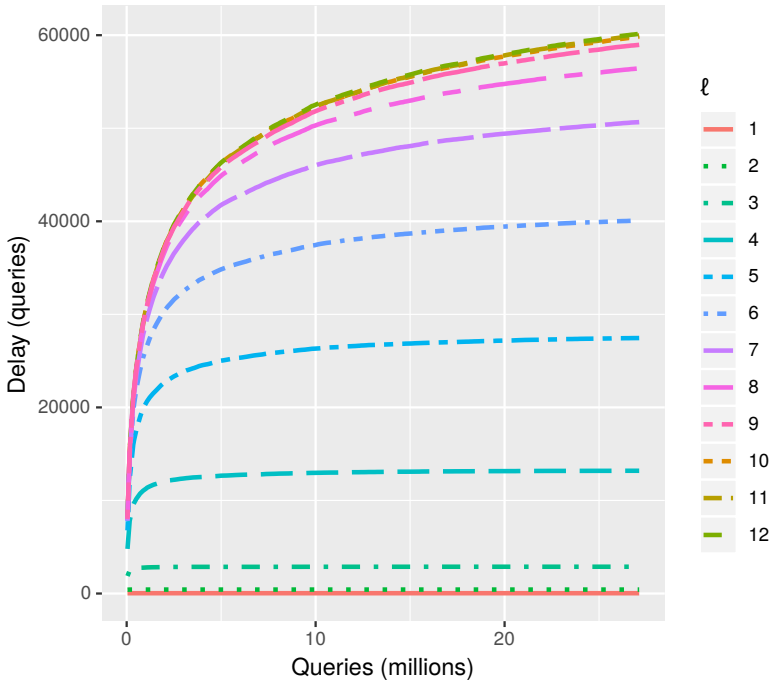

FIGURE 4.16: Queries delay, as the mean number of other queries that enter the system during the period between the entry and the leave of a given query. Once the categories are full, the output stabilizes.

4.4.5.5 Resource Consumption

Notice that our algorithms do not use any disk space, therefore only memory consumption needs to be evaluated.

We have identified the variations in ℓ-value as the main parameter that affects resource consumption. Memory consumption increases when a new level of depth is added to the tree, in proportion to the number of effective categories that are added (cf. Table 4.4). Categories were created dynamically, depending on query's classification, therefore a different data set will generate different categories. At the end of our tests, we used a maximum of 194 505 categories, in a tree with depth thirteen.

With our test data, we see that most records are classified at depths between five and seven, although we found a maximum depth of thirteen. As we increase depth, there are fewer queries that can be classified at the last levels, using the same data and the same classifier. Although we increase the value of ℓ the effective

ℓ	**Added categories**	**Total categories**
1	16	16
2	537	553
3	5 523	6 076
4	21 786	27 862
5	36 806	64 668
6	35 543	100 211
7	39 998	140 209
8	26 914	167 123
9	16 863	183 986
10	7 863	191 849
11	2 143	193 992
12	441	194 433
13	72	194 505

TABLE 4.4: Number of categories added with each increase in ℓ-value and total categories of a tree with ℓ depth. Although we found a maximum depth of thirteen, we see that most records are classified at depths between five and seven.

number of categories created is marginally increased from this point. This also causes memory consumption to stabilize. Let us illustrate the previous observation with an example. Given a query classified as "a:b:c:d:e" if we use an ℓ equal to 4, the level 4 vertex "a:b:c:d" is used for anonymization. If we increase ℓ to 5, or a higher value, we use for anonymization the complete category, i.e. level 5 vertex "a:b:c:d:e", even if we use an $\ell = 13$.

On the other hand, we can see that k adds a multiplicative factor in the consumption of resources, depending on the number of existing effective categories. The results in Figures 4.17, 4.18 and 4.19, only show the maximum memory consumption.

Regarding different algorithms set forth, both *anonymizer* and *de-anonymizer 1* show the same memory consumption profile. *De-anonymizer 3* is the algorithm with higher memory consumption. This is because that algorithm creates user profiles in memory and therefore is reasonable that it uses more resources. *Anonymizer* and *de-anonymizers 1* and *2* should not use more memory than the reported, regardless of the volume of logs they deal with. However, this is not

the case of *deanonymizer 3*, as when it creates new user profiles, it increases the memory consumption.

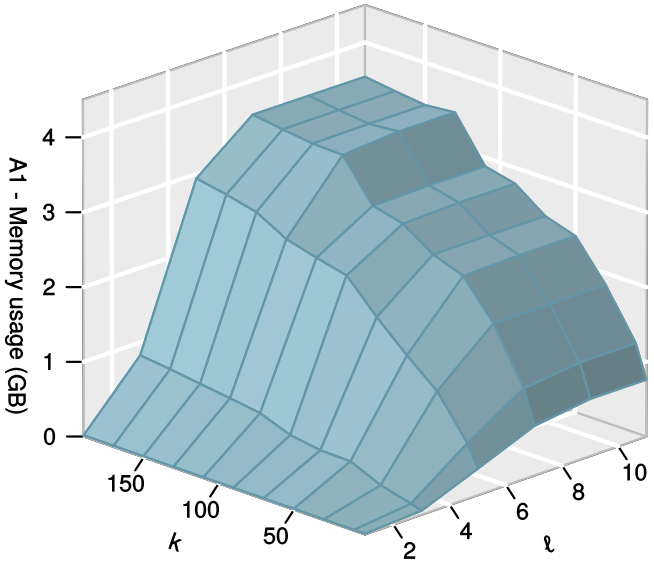

FIGURE 4.17: Anonymizer/De-anonymizer 1 - Memory consumption (GB). The value of ℓ is the main parameter that affects memory consumption. The value of k adds a multiplicative factor. Both the *anonymizer* and *de-anonymizer 1* show the same memory profile.

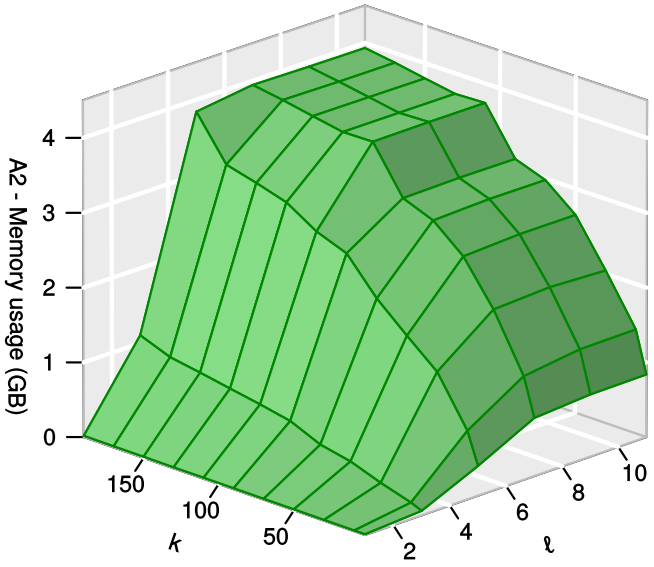

FIGURE 4.18: De-anonymizer 2 - Memory consumption (GB). In this case, the value of ℓ is also the main parameter that affects memory consumption, and the value of k adds the same multiplicative factor.

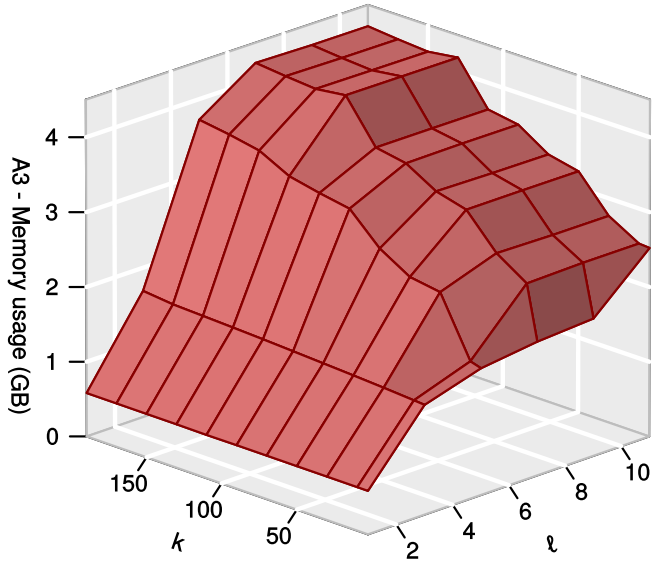

FIGURE 4.19: De-anonymizer 3 - Memory consumption (GB). The value of ℓ and k affect memory consumption in the same way. However, in this case we have a higher memory consumption, because it creates user profiles in memory.

4.4.5.6 Efficiency

As we have seen in the previous sections, a lightweight method has been defined. It allows the logs to be quickly processed with reduced resource consumption.

Studying the anonymizer we see that both delay and memory consumption vary initially, because the system starts empty and the sets must be filled. As we have seen, once the sets achieve k elements, these values stabilize. On the other hand, the processing speed of a log depends on the value of k and ℓ, but it remains constant throughout each test set.

Analyzing the proposed algorithm, we can see that each log is only treated once. This allows us to equate its efficiency with well known singly-linked list traversal algorithms. Therefore, the algorithmic time complexity of our proposal is linear regarding to the input and could be established as $\mathcal{O}(n)$.

4.4.5.7 Transparency

The input of the system should be a stream of classified query logs that can be obtained from the WSE. In case that only unclassified logs are available, a classification micro-service could be implemented and added to the WSE architecture, as we previously showed in Chapter 3. In case that classified logs are available, those logs could be used without further modifications. Our system generates an anonymized stream of logs, preserving the existing structure. From the point of view of an existing client, generated output will be completely indistinguishable of the original one. Therefore, total transparency is reached.

4.5 Conclusion

A formal approach for the anonymization of WSE query logs has been presented. Our proposal allows to publish query logs without any other modification than eliminating direct identifiers and equivalent user re-assignment categories. This contrasts with existing approaches that release heavily modified data, either distorted or generalized, to maintain anonymity. In addition, our proposal allows some degree of configuration, using two main parameters:

- k to adjust the level of diversity on each category.
- ℓ to adjust the amount of available categories.

This parameterization allows to adjust privacy and utility levels of generated logs according to the needs of each application.

Three algorithms have been evaluated performing an attack to the anonymized data, using the most favorable scenario for the attacker, i.e., when the attacker knows the algorithms used by the WSE, all the parameters and the data. The attacker has access to the anonymized log stream, but not to the original logs. Tests with this context and several values of k and ℓ were conducted.

Our best record-linkage attempt re-identified 23.18% of original logs with the lowest k-value, highest ℓ-value and using the most complex record-linkage algorithm, which is also the one that needs more resources. With the same parameters, using the simplest record-linkage algorithm we get an 18.36%. These results are reduced rapidly, recovering less than 1% of original logs when using values of k over 100. Variations in the values of ℓ do not have a representative impact in terms of record linkage, but they do offer a significant improvement in terms of data utility.

Our proposed ideas were tested using the *AOL* released logs, showing the feasibility of our solution over real environments. The application of our work is sufficient to generate anonymized logs that meet representative criteria, e.g., release of anonymized data to third parties. Our solution can handle the equivalent to *Google*'s average load, using only one execution thread in our testing environment. To evaluate log's utility after anonymization, we have measured distances between user profiles using Earth Mover's Distance. We have found that using an ℓ-value of one, a 42.03% of utility was lost. Using ℓ-values of six or more, less than 1% of utility was lost.

Chapter 5

Lifelogging Protection Scheme for Internet-based Personal Assistants

5.1 Introduction

Traditionally, people have used Web Search Engines, as the main gateway to the Internet, but as time goes by, new proposals are trying to reduce the barriers to access information even more.

In a broad sense, most of these new proposals are considered Smart Things [157], a group which includes smartphones, smartgateways, smartwatches and activity bands. A subset of Smart Things is getting a lot of attention in recent years. This set of devices is known as Internet-based Personal Assistants, containing some outstanding examples, such as Google Home and Amazon Echo [5, 44].

The traditional use of the WSEs is expected to be broaden when using Personal Assistants, to aid domestic users in their household activities: shopping, travel planning and home automation, among others. The remote participation of WSEs and Social Network providers in the process, makes it already possible to process

vast amount of information to increase social experiences such as scheduling of meetings and spontaneous gatherings with family members and friends [44].

It is often easy to forget that when we interact with some online services, our usage data is stored on the Internet. Using Personal Assistants, since the user is not in front of a computer, the situation is even more accentuated. Users tend to establish more relaxed relationship with the device, sometimes without even knowing if it is working or sending information to another site, and mostly seeing it as a friend or an extension of its person.

The recording and processing of all this information, denoted as lifelogging, that is a term used to describe recording our everyday lives. Current applications of lifelogging are built around remembrance or searching for specific events from the past. In [7], they propose to extend this to allow them to characterise and measure the occurrence of everyday activities of the user and in so doing to gain insights into the user's everyday behaviour. Their methods are to capture everyday activities, and to use an algorithm which indexes the occurrence of basic semantic concepts. Then use data reduction techniques to automatically generate a profile of the user's everyday behaviour and activities. They conclude that the overall performance of these techniques makes it usable for characterizing the lifestyle and behaviour of subjects. Therefore, it is possible to generate a more enriched users' profiles, using all the data generated by these devices.

Related to Internet-based Personal Assistants' architecture, novel privacy risks also arise. In [4] current topics in Smart environments are discussed while describing interconnection issues, security threats and suggesting a lightweight framework for ensuring security, privacy and trustworthy lifelogging. Under the scope of the presented framework, new issues and security threats were undercovered. Security challenges appear due to the lack of suitable security mechanisms and protocols in the Internet of Things because of the limited resources of smart objects.

To sum up, in this chapter, we address the issue of transforming raw user's data from lifelogging data streams generated by Internet-based Personal Devices like Google Home and Amazon Echo. We study the relation of such devices with other

data information actors in terms of EU data protection directives and propose a protection solution via anonymity transformation. Our proposal takes into account the role of the organizations and their needs to monetize generated data. Our protection scheme aims at limiting the risk of privacy disclosure, while maintaining an adequate level of data utility.

The remainder sections of the chapter are organized as follows. Section 5.2 introduces our problem statement. Sections 5.3 and 5.4 present the general overview and architecture of our proposal. Section 5.5 concludes the chapter.

5.2 Problem Statement

5.2.1 EU Data Protection Actors

EU Directive 95/46/EC [158], nowadays superseded by the new General Data Protection Regulation (GDPR) [38], defines different roles that are relevant to the protection of general-case lifelogging environments. First, it defines the *Data Controller* as *the natural or legal person, public authority, agency or any other body which alone or jointly with others determines the purposes and means of the processing of personal data.*

Lifelogging environments need to clearly identify who is the *Data Controller*, since it determines which national law is applied. The data controller is the responsible for determining what data must be processed, which third parties can access this data and when this data must be deleted.

In addition, the figure of the *Data Processor* has the responsibility to ensure the security in the processing of personal data. The directive states that it is the *natural or legal person, public authority, agency or any other body that processes personal data on behalf of the controller.* It is also necessary to determine the *Data Processor*, as it also sets the national law to be applied. It is also necessary to consider the *Data Subject*, as the person who is generating the data and from

which we need the consent. The directive also requires to guaranteeing a set of basic rights to the *Data Subject*, such as the right to access their information or to oppose to the data processing.

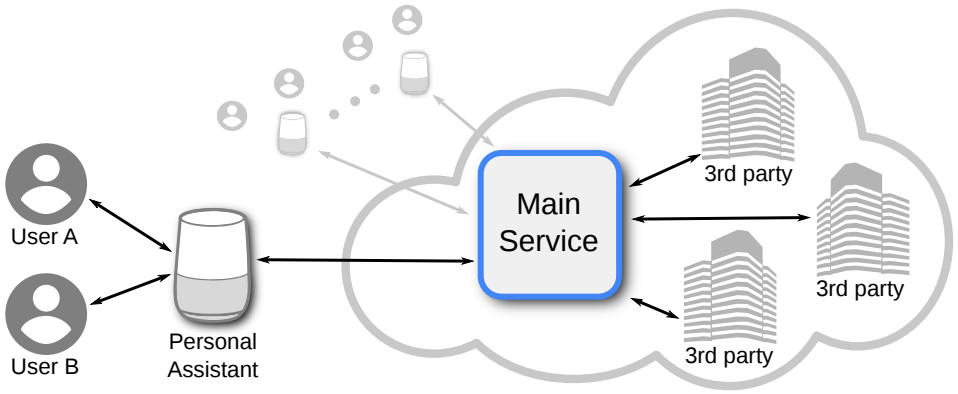

FIGURE 5.1: Main Architecture. *Users* represent the data subject, and are authorized to interact with the *Personal Assistant* devices, by submitting queries and commands. *Personal Assistant* devices then send those commands to the *Main Service* that take the role of data collectors. Finally, the *Third Parties* are the entities acting as data processors. They represent those parties that express interest on legitimately accessing the anonymized query and command logs.

Figure 5.1 depicts a lifelogging environment which involves several actors, namely: *Users*, *Personal Assistant devices*, *remote Main Services* and *Third Parties*. Users represent the actors related to data subject, i.e., they represent the entities that are authorized to interact with the Personal Assistant devices, by submitting queries and commands. The Personal Assistant devices receive both queries and commands from associated users. Queries and commands are sent and processed by the Main Services for customized results. The remote Main Services take the role of data collectors. They have direct access to the original queries and, e.g., command and control logs, sent by the Personal Assistant devices. Third Parties are the entities acting as data processors. They represent those parties that express interest on legitimately accessing the anonymized query and command logs, to eventually process and use them.

TABLE 5.1: General search query example

Request	What's the most populous country in the world
Response	Ten Countries with the Highest Population in the World - Internet World Stats
Time	25 Apr 2018, 13:34:37
Products	Assistant
Locations	48.624924, 2.443970

TABLE 5.2: Location based query example

Request	What's the weather like tomorrow?
Response	In Paris, tomorrow, there will be storms intermittently, with a maximum temperature of 24 degrees and a minimum temperature of 13 degrees
Time	22 May 2018, 14:09:53
Products	Assistant
Locations	48.624924, 2.443970

5.2.2 Data Structure

Personal Assistant devices may receive three different types of queries: (1) general search queries, (2) location based queries and (3) commands. They are transferred to the Main Servers for processing. Hence, the Main Service stores all the original logs for each Personal Assistant device with respect to its different associated Users. Queries and commands are defined as follows:

- **General search queries** — These are the traditional queries we are accustomed to send to a Web search engine using, e.g., a computer or a mobile phone. These queries help users to find what they are looking for, from over 1.8 billion websites [159]. Users just have to ask a question and the system returns the main result they are looking for. The query example 5.1 illustrates a general search query log, based on the Google-Home personal assistant device.

- **Location based queries** — based on spatial and temporal data, location-based queries are classified on two categories: elementary queries and derivative queries. *Navigation and search for Point of Interests* are typical elementary location based queries. Derivative queries are mainly processed for *guiding* or *tracking* to provide customized results to users. The query example 5.2 presents an elementary location-based query log, based on Google-Home personal assistant device.

TABLE 5.3: Command example

Request	Can you increase the sound to 20%
Response	
Time	3 May 2018, 09:06:01
Products	Assistant
Locations	48.624924, 2.443970

- **Commands** — They allow users to request direct actions that affect their own environment. These actions are usually related to home automation, multimedia control, alarms, lists, etc. Although these actions usually only have a local repercussion, all the data they generate is also stored together with the rest of the logs. The example 5.3 presents a command log, based on Google-Home personal assistant device.

5.2.3 Security and Functional Requirements

The proposed scheme has to fulfill a series of security and functional requirements. First of all, and in terms of **user privacy** entities' privacy can be preserved thanks to the query de-identification, using different approaches, namely full de-identification, partial de-identification and statistical de-identification. It is commonly agreed that statistical de-identification is the more suitable approach. It is accomplished trough many statistical criteria and anonymization techniques, being k-anonymity and its extension l-diversity the two most widely accepted techniques. Those techniques were proposed for structured data [78, 79, 85, 86]. In terms of **scalability**, the system shall be capable of handling any increase of workload. It shall be capable of accommodating any type of growth [149]. As an additional requirement, the system shall guarantee that the **resource consumption** is kept low, to facilitate its inclusion in existing architectures, minimizing overhead cost. Even though the system is supposed to scale with added resources, it is important not to increase unreasonably the current system's resource consumption.

In terms of **data utility**, and although full de-identification is desirable from a privacy standpoint, organizations need to monetize some non-sensitive user information, selling it to a third party. Also third parties want to extract some useful

user characteristics from the data they bought. The main problem is that query and command logs can also reveal sensitive information, raising some serious privacy concerns. So there is a trade-off between anonymizing logs and keeping them useful to extract information through data mining processes. Therefore, the main challenge related to data utility is to anonymize sensitive user data removing as few information as possible in order to have enough interesting information to be analyzed. To do so, the proposed scheme aims to build synthetic logs and user profiles, which should maintain users' interests and break quasi-identifiers that could allow to identify an user. Queries should be anonymized to not relate sensitive information to a user identity. It should be as difficult as possible to relocate queries in order to build original user's profile. In the end, the proposed system should generate those fake logs and profiles with other users' queries.

5.3 Proposal

In this section we will propose a preliminary implementation of a system to protect users' privacy, protecting logs generated from the use of virtual assistants. This proposal wants to be a proof of concept to study the viability of this type of solutions and promote its discussion.

Our ultimate goal is to allow the monetization of those logs by the organizations, preserving the privacy of users. This will also allow these logs to be sold to third parties, without disseminating sensitive user information.

As we have seen, working in a trusted environment may have some problems. In our proposal we assume that we are in a semi-trusted environment, where all the actors can be honest but curious.

To achieve a viable architecture in this environment, we need to add some actors to the basic architecture of the system, as we can see in Figure 5.2.

The Identity Screener ensures the compliance with the legal constraints and requirements to settle, e.g., privacy prevention algorithms, based on criteria set by

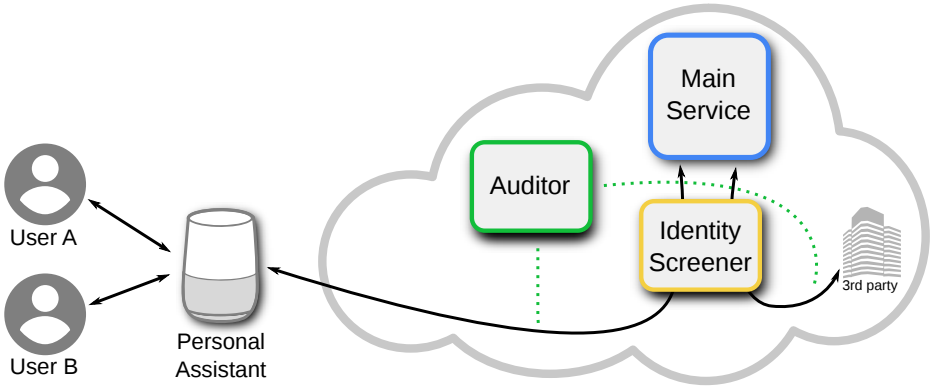

FIGURE 5.2: Proposed architecture for semi-trusted environment, where all the actors can be honest but curious. It contains a Identity Screener which acts as a container of privacy algorithms, and an Auditor, responsible of auditing accountability and users' consent requirements.

EU regulation directives. Indeed, it acts as a container of privacy algorithms to enforce data protection and control any misuse between the other parties. In the following section, we describe more in depth the working properties of an idealized Identity Screener conducting sanitizable signatures and log anonymization. The Auditor is a dedicated entity which is responsible of auditing the Identity Screener and Main Service activities with respect to accountability and users' consent requirements.

5.3.1 Sanitizable Signatures

The notion of sanitizable signatures was introduced in [160]. Sanitizable signatures are malleable mathematical schemes, that allows an authorized semi-trusted party, referred to as *sanitizer*, to modify designated portions of a signed message in a limited and controlled fashion.

The sanitizer can modify parts of the original message m when the original entity that created the message, referred to as the *signer* provides a description of the admissible modifications, hereinafter denoted as ADM, for each message-sanitizer pairs. That is, the signer divides the message $m \in \{0,1\}^*$ into N blocks $m_1, \cdots, m_N$, defines the set $\text{ADM} \subseteq \{1, N\}$ of admissible blocks and signs the whole message using a key related to the sanitizer. Using the aforementioned

key, the sanitizer is able to modify the admissible parts of m in a way that keeps the resulting signature still valid, under the public key of the signer. Therefore, the sanitizer could produce a valid signature of the legitimately modified message without interacting with the original signer.

There are several constructions for this primitive, based on standard signature schemes, such as chameleon hashes, which are proved secure under common cryptographic assumptions.

Sanitizable signatures can also be improved to satisfy *unlinkability* [161]. Unlinkability is a strong privacy property which forbids a third party from linking two (or more) messages. In other words, it guarantees that is unfeasible to distinguish between sanitized signatures that have been produced from the same message or by the same sanitizer. It is also possible to limit the set of all possible modifications on one single block and how to enforce the same modifications on different messages blocks [162].

The aforementioned properties, draw sanitizable signatures as specially interesting for architectures where multiple actors interact, such as in the Personal Assistant environment. The implemented scheme and experimental performance measurements for the implementation of a sanitizable signature scheme, demonstrates that the scheme could be practical and efficient for our application.

5.3.2 Probabilistic k-anonymity

Most approaches that want to anonymize data in real-time use k-anonymity and try to minimize the time needed to process the data [87, 88]. A data-set that has been k-anonymized, fulfills the property that each record is indistinguishable from at least $k-1$ other records, and therefore no user can be re-identified with a probability greater than $\frac{1}{k}$, using only record linkage attacks. We believe that a more appropriate approach may be based on probabilistic k-anonymity [91], which relaxes the indistinguishability requirement of k-anonymity and only requires that the probability of re-identification be the same as in k-anonymity. This approach

may be suitable for Personal Assistant logs as it is adaptable to a streaming environment. In addition this process is also very fast in one way, anonymization, but difficult to undo. Probabilistic k-anonymity also allows to select desired user privacy level modifying k parameter values. The other advantage of using this approach is that anonymized logs are generated using real user queries, they are not modified, but distributed among other users with similar interests. In this way the quasi-identifiers get dispersed between several users and thus prevent record linkage attacks, but data utility can be preserved as well (see Chapter 4).

5.4 System Overview

We propose the creation of a modular system, following the design that can be seen in Figure 5.3. The system consists of five different parts. Within each of these parts we divide the tasks into different modules. Each module will be responsible of a single action in order to accomplish the *Single Responsibility Principle*, defined as a micro-services architecture.

Our proposal is focused on the Request Architecture. To have the full system operative, it is necessary to add more components to that architecture to handle the responses, using the same concepts. Therefore, the proposed system begins with the interaction of the User and the Personal Assistant, which will generate a set of queries. These queries will be sent through the network for treatment. Once treated, the resulting logs will be properly anonymized, increasing the privacy protection of the user. That makes possible to sell these protected logs to third parties to monetize them. Below we will briefly describe the functions performed at each stage of the proposed system.

5.4.1 System Initialization

As a prior step to the start of system execution, we must ensure the distribution of the key pairs to create and check the User Sanitizable Signatures and the Service

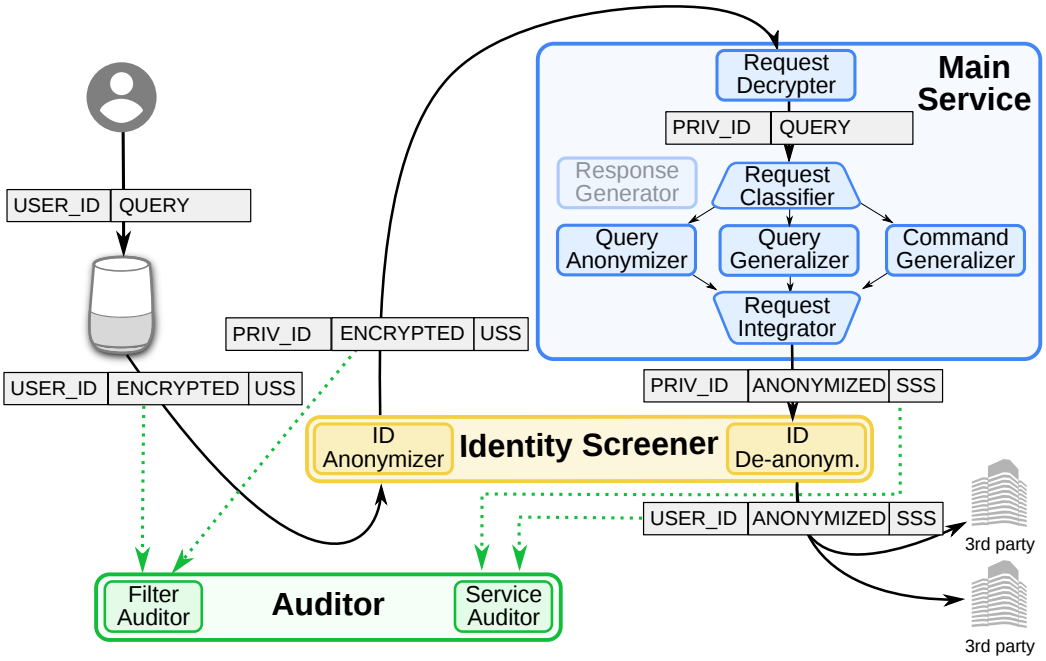

FIGURE 5.3: Request Architecture. The User will interact with the Personal Assistant, which will generate a set of queries. These queries will be sent through the Identity Screener, which will hide the user identity. Then the Main Service will treat and anonymize these queries. After passing through the Identity Screener again, it is possible to safely sell these protected logs to Third Parties.

Sanitizable Signatures, as well as the certified public key of the Main Service to all the Personal Assistants.

5.4.2 Query Pre-processing

The pre-processing steps are performed before the query reaches the Main Service. These steps encrypt the query, and replace the original user ID with an anonymous one.

5.4.2.1 Local Device

The user sends a question to the Local Device, that recognizes who has formulated it and transforms it into text. Once transformed, our model proposes to encrypt this query using the public key of the Main Service and then sign it using the User

Sanitizable Signature. This signature will allow the Identity Screener to modify the user's part, but not the rest of the message.

5.4.2.2 ID Anonymizer

Then the query is sent to the Identity Screener, which is a distinct administrative entity than the Main Service. Using the ID Anonymizer module, it is responsible for replacing the original USER_ID with an anonymous ID, which we call PROX_ID. The purpose of this change is to prevent the Main Service from knowing the real identity of the user that generated the original query. We want to emphasize that Identity Screener do not have access to the original query, since in this point is encrypted.

5.4.3 Anonymization

The anonymization process, which is conducted entirely in the Main Service, is the responsible of anonymize and generalize the requests. All steps in this part, are carried out without knowing the real identity of the users, thanks to the preprocessing steps.

5.4.3.1 Request Decrypter

The first step that the Main Service does is to verify that the signature of the query is correct. Then it discards that signature and proceeds to decrypt the body of the query with the Main Service private key.

5.4.3.2 Request Classifier

As introduced in Section 5.2.2, our system will receive three different types of logs. The main responsibility of the Request Classifier is to process all the logs, to find the type of each log and to decide how it will be treated. Therefore, it acts

as the facade or distributor part of the system. By itself it does not make any modification to the data generated by the user.

5.4.3.3 Query Anonymizer

This module is the responsible for anonymizing general search queries. To do this, we propose to use a system based on probabilistic k-anonymity (Chapter 4). This system uses a natural language categorizer of logs in real time and for each category creates two sets of k elements. In the first set, it stores the users and in the second set the queries. When a category contains k elements, the algorithm randomly picks one user and one query from each of the sets in the same category. With them, it builds a new anonymized log, with a minimum delay. This system obtains a level of protection similar to other methods using k-anonymity.

5.4.3.4 Query Generalizer

This module will be in charge of anonymizing the Location Based Queries. To do this, we propose to use an approach similar to the following [163–165]. In this case, the situation is quite different, since both the query, all the associated information (location and time) and the response that the user receives, may disclose something that affects directly the user's privacy. The basis of the system is also k-anonymity, but also some data perturbation system may be applied, for location and time information. It is difficult to unleash the logs of that effect, since the results with better utility are the ones that are directly related to the users location and time. This raises new challenges when it comes to protecting this information, without losing data utility. Therefore it would be desirable to be able to configure this module with different levels of privacy/utility, depending on when and where the query was generated, in order to protect the identity of the user.

5.4.3.5 Command Generalizer

We will use an algorithm similar to the one proposed in the previous section, to be able to obtain anonymity on the user commands, as well as to anonymize both the moment and the exact location of the user that issued them. We deal with the commands in a different way, than with the queries, as we anticipate that we will face different requirements when it comes to reducing the risk of information disclosure in both scenarios.

5.4.3.6 Request Integrator

Once we have performed all the anonymization tasks in the previous modules, this is the module in charge of unifying the results and then generating and publishing the logs as the main system output. It also adds a Service Sanitizable Signature, which will allow the Identity Screener to modify the user field, but not the rest.

5.4.4 Query Post-processing

The post-processing, in the current version, is only performing the ID De-anonymizing step, which is mandatory for the proposed architecture. However, further steps may be included in this part if required by third parties.

5.4.4.1 ID De-anonymizer

Once the Main Service has finished performing all of its tasks, the resulting logs are resubmitted to the Identity Screener which, in turn, checks the Sanitizable Signature Service and if it is correct then proceeds to restore the original USER_ID, through the ID De-anonymizer. We need to do this, since the third-party pays for this relationship between the user and his interests. However, it should be noted that now the text of the query is conveniently anonymized and therefore, despite being able to extract the interests of users, they can not perform record linkage attacks to obtain sensible information that threatens user privacy.

5.4.5 Response

Below we will explain how the response is sent to the user. It is not the main focus of our proposal, but we think that it is necessary in order to fully understand the proposed architecture. An overview of the response architecture can be seen on Figure 5.4.

5.4.5.1 Response Generator

Once the request has reached the Main Service, and passed through the Request Decrypter, the Main Service has access to the user's original query. Therefore it can proceed to search for the necessary information and create the right answer for the user, using Response Generator. Once this response has been created, the same Response Generator will encrypt it and sign it with a Server Sanitizable Signature.

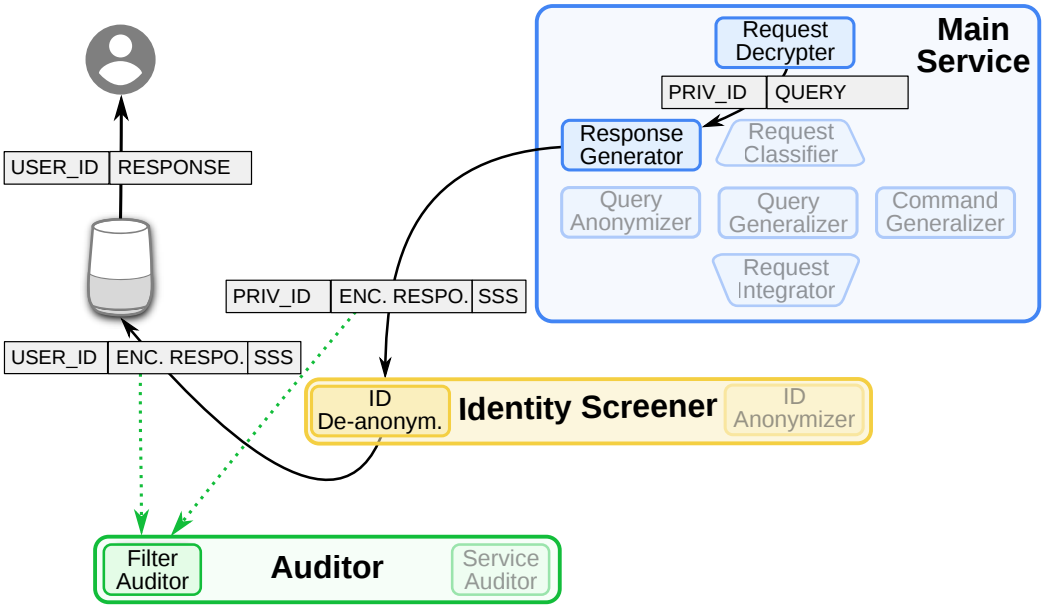

FIGURE 5.4: Response Architecture. The Main Service creates and signs the response for the User, using Response Generator. Then, it is submitted to the Identity Screener, which restores the original USER_ID, through ID De-anonymizer, and send it to the Personal Assistant device, which decrypts and provides the result to the user.

5.4.5.2 ID De-anonymizer

Once the Main Service has finished performing all of its tasks, the resulting record is resubmitted to the Identity Screener, that checks the Sanitizable Signature Service. If this is correct, then the Identity Screener proceeds to restore the original USER_ID, through the ID De-anonymizer. However, it should be noted that the response text is conveniently encrypted. Once this response reaches the Personal Assistant device, it will be decrypted and provided to the user.

5.4.6 Audit

The auditing process is performed by a dedicated authority, mainly relying on the verification process of Service Sanitizable Signature. That is, the auditor has to verify the consistency of signed queries and responses, generated by the User, the Identity Screener and the Main Service, such as:

- **Identity Screener activities auditing** — the auditor is able to verify the consistency of signed queries, and responses, generated by the Identity Screener. In other words, honestly generated (i.e., *signing correctness*) and sanitized (i.e., *sanitizing correctness*) signatures have to be accepted by the verifier, and honestly generated proofs on valid signatures (*proof correctness*) have to be accepted by the judge algorithm of Service Sanitizable Signature [160].

  Recall that during the request process, the Identity Screener sanitizes the user identifier using the PROX_ID before sending the sanitized generated query to Main Service. Consequently, a correspondence table should be maintained by the Identity Screener, to be able to recover the USER_ID, when sanitizing the response query received later by the Main Service. In addition, to comply with GDPR [38], with regard to accountability purposes, the Proxy Filter has to store a proof for every personal data processing. That is, for each USER_ID, a set of signatures' tuples associated with a session identifier needs to be stored. Each tuple is represented as follows:

{session identifier, signed query (PA), sanitized query (Identity Screener)}

Hence, the auditor may check both generated signatures, by the Personal Assistant and the Identity Screener during the request process, and by the Main Service and the Identity Screener, during the response process and also compare consistency between the original signature and sanitized one.

- **Main Service activities** — similarly, the dedicated auditor verifies the consistency of signed original queries' responses and anonymized query logs, generated by the Main Service.
  Recall that the Main Service should store anonymized query logs and is able to process personal data, only with the consent of the corresponding data owner [38]. That is, when registering with Main Service, each user generates an authorization vector `av`, specifying the allowed actions on his data. For instance, for transfer actions, each user has to specify which data processors (Third Party) are authorized to collect his data from the Main Service. As detailed below, to comply with GDPR [38], with regard to user consent compliance, Main Service has to store a proof for every personal data processing. For instance, for each transfer process, a set of signatures' tuples associated with a session identifier needs to be stored. Each tuple includes the transfer session identifier, the original signed query, the anonymized signed query as well as the third party identifier. It is represented as follows:

  {transfer session identifier, signed original query (MS), signed anonymized query (MS), TP identifier}.

  As each anonymized query has to be sent the Proxy Filter, in order to sanitize the query identifier using the USER_ID, before transmitting to Third Parties. Hence, the auditor may check both generated signed and sanitized signatures, by Main Service and the Identity Screener respectively and also verify if transfer actions are allowed with regard to each user authorization vector.

5.5 Conclusion

Internet-based personal assistants can lead to serious privacy risks. They can release sensitive information about the identity of domestic users and their sensitive data. The issue must be tackled by jointly addressing anonymization by organizational roles in terms of *Data Controller*, *Data Processor* and *Data Subject*. Towards this end, we have proposed an architecture that combines lifelogging anonymization and sanitizable signatures, to promptly mitigate privacy threats.

We believe that the proposed architecture is very encouraging, in order to define a system that allows us to work in a trusted-but-curious Internet-based Personal Assistants environment. Anyway, we are aware that this approach has some limitations in which we are still working. Regarding the part of communication with the user, it must be ensured that the Personal Assistant does not send information to the Main Service directly, therefore escaping the treatment of the Identity Screener.

Regarding the communication with the Third Parties, we do not have this problem, since if they want to recover the original USER_ID, all messages must go through the Identity Screener. In this case, the possible privacy problem would appear if any of the third parties send the data back to the Main Service once it have been processed by the Identity Screener. In this case, the Main Service, would have access to the anonymized query and the original USER_ID. Therefore, if the Main Service has saved the correspondence between the original query and the anonymized query, it could fetch the original query and user pair.

We are currently in a preliminary work stage of this system, with the limitations that this implies, but we are trying to improve this aspect of our proposal.

Next steps include a more thorough analysis about the cooperation of the different elements of our architecture, as well as to provide further investigation about the effective techniques included in the architecture with a specific brand of Internet-based personal assistants. Snippets of code, available at http://j.mp/lps-ipa,

based on a cryptographic library [166], released to facilitate the understanding and validation of the proposed architecture.

UNIVERSITAT ROVIRA I VIRGILI
CONTRIBUTIONS TO LIFELOGGING PROTECTION IN STREAMING ENVIRONMENTS
David Pàmies Estrems

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In order to access the Internet people need to use search services, whether this be in their classical Web Search Engine form or through personal assistants, which provide a more transparent tool for accessing information. Interactions with these services leaves a record of interests, personal and work information. For service providers, these records are very valuable, since they provide a way to improve the service, and a way to monetize it.

Monetization is achieved by selling data to third parties, but it is necessary to be extremely careful in the process. Selling data may generate huge negative consequences, since query logs may contain identifiers or sensitive data from users (such as diseases, sexual tendencies, religious beliefs). Besides ethical considerations, current regulations oblige companies to protect personal information.

In the literature review, protection systems for data-sets were found, most of them working with perturbative approaches over atomic files. Unfortunately, the scenario posed by Internet searches is closer to a real time stream of the user's data. Considering this scenario, we did not find an ideal solution for these requirements, given that the data is intended to be consumed once protected, and therefore have to maintain a high degree of utility.

In this sense, a novel framework for anonymizing query logs has been presented. Privacy guarantee is defined in terms of set theory, which relates sets of users to sets of query logs. Data could be released without modifications other than removing direct identifiers from query text and remapping between those two sets. This contrasts with existing approaches that release heavily modified data, either distorted or generalized, to maintain anonymity.

Regarding privacy, we have devised formal and experimental proofs to ensure proposal's comportment. In our evaluations we have considered the worst-case scenario, in which an attacker who is willing to use the anonymized query logs to retrieve the original ones has gained access to the base information of the service: algorithms, anonymized logs, k and ℓ values.

Our first proposal is fast and with a high level of privacy. However, since the number of categories it uses is low, the utility of the anonymized data could be improved. This led us to our second proposal. In this case, the proposal allows one to adjust the amount of available categories, as well as the number of elements in each category. This parameterization permits different privacy and utility levels, according to the needs of each application.

In order to evaluate query log's utility after applying anonymization, we have measured distances between user profiles using Earth Mover's Distance and also manually classified and compared a sample of logs. To achieve comparable results between our first and second proposal, we used the second one with an ℓ-value of 1, which draws them equivalents in terms of anonymization. Studying the results we have found that just above 40% of utility was lost in both cases. However, utility increases rapidly with bigger ℓ-values in our second proposal. With ℓ-values of 6 or more, less than 1% of utility was lost, without affecting privacy, since we achieve a protection of $\frac{1}{k}$ in all cases. Therefore, the application of our proposal is sufficient to generate anonymized logs that meet the defined utility criteria and could be released to third parties safely.

Algorithmic time complexity of our proposal is linear regarding to the input and could be established as $\mathcal{O}(n)$. We have also considered the average Google's load,

in queries per second, to study the run-time cost and the memory usage. The results show that proposed anonymizer algorithms could handle the equivalent to the average of Google's load in real-time, using only one thread of execution and a feasible amount of memory, in our test environment.

A full architecture, taking into account the use of Internet-based Personal Assistants, was finally presented. This architecture, jointly addresses anonymization by organizational roles in terms of *Data Controller*, *Data Processor* and *Data Subject*, in order to comply with the guidelines of the GDPR. Arriving at the final proposal of an architecture that combines lifelogging anonymization and sanitizable signatures, to promptly mitigate privacy threats.

6.2 Future Work

The proposed architecture is very encouraging in defining a system that allows us to work in trusted-but-curious Internet-based environments. We are aware that this approach has some limitations, but there are some parts that may be further expanded on and become future work lines.

All the proposals that have been presented need to use classified queries in order to later protect them. In our case, we used a custom classifier, which was not the main subject of our research. The proposed classifier algorithms could be the subject of further study. The classifier may be improved by means of a more accurate natural language analysis in order to perform a semantic analysis of queries. However, the classifier was also the slowest part of the process and a more sophisticated one may need more processing power. Anyway, there is room to develop more efficient alternatives using NLP or Artificial Intelligence (AI) systems to perform query analysis.

AI by means of the subfields of machine learning (ML) and search, provides some techniques that can be used to deanonymize transformed datasets. Impact on anonymity can get higher when data combines merged information from heterogeneous nature, data sources coming from third parties which, although not malicious, may have different objectives from those of a defender (e.g. sharing anonymized or truncated data that becomes unusable or even harmful). This concerns the protection against attacks on the learning phase, which we consider as an interesting complementary line of research.

In order to properly control the risk, attacks on the test phase must also be addressed. Regarding this last point, the problem can be summarized by the fact that a very precise classifier will have difficulty generalizing his decisions and conversely a classifier able to handle data relatively different from those on which he has been trained will tend to be quite imprecise. Overly precise algorithms will be difficult to generalize but will also be more sensitive to attacks resulting in

poor classification on slightly noisy data [167]. Conversely, an algorithm that is sensitive to detail will tend to let malicious content flow more easily.

We propose to integrate this kind of compromise into the security game modeling risk management. An interesting avenue would be to consider the possibility for a classifier not to produce an answer. This behavior corresponds to the rejection principle, which is relatively well known in the classification community, but to our knowledge, is poorly understood in the context of application to security [168].

Regarding the anonymizer, our best proposal is able to work with custom degrees of classification. However, the degree, once selected, becomes fixed. It would be interesting to expand the anonymizer to use dynamic degrees of classification, both globally or for some specific categories.

Although the proposed architectures use a micro-service approach, a distributed environment remains untested. A more thorough analysis of the cooperation of the different elements would be beneficial. This could improve the system performance, but other concerns may arise in terms of management, security, and privacy.

When we take into account the use of Personal Assistant devices, as the main communication interface with the user, it must be ensured that these devices act honestly. In fact, they should not have the opportunity to send information directly to the Main Service, therefore escaping the treatment of the proposed Identity Screener, however, this could be addressed by a simple address control at the local router.

Regarding the communication with Third Parties, we do not have this problem, since if they want to recover the original USER_ID, all messages must go through the Identity Screener. The possible privacy problem would appear if any of the third parties send the data back to the Main Service once it has been processed by the Identity Screener. In this case, the Main Service would have access to the anonymized query and the original USER_ID. Therefore, if the Main Service has

saved the correspondence between the original query and the anonymized query, it could fetch the original query and user pair.

Finally, a large scale investigation with a specific search engine and a brand of Internet-based personal assistants would provide a full validation of the proposal.

Bibliography

[1] David Reinsel, John Gantz, and John Rydning. Data age 2025: The digitization of the world from edge to core, an idc white paper, November 2018. URL https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf.

[2] Wikipedia. Web search engine. https://en.wikipedia.org/wiki/Web_search_engine, 2020.

[3] C.R. Tris, Universitat Rovira i Virgili. Departament d'Enginyeria Informàtica i Matemàtiques, and Universitat Rovira i Virgili. Escola Tècnica Superior d'Enginyeria. *Client-side Privacy-enhancing Technologies in Web Search*. Sescelades: Tesis. Universitat Rovira i Virgili, 2014. URL http://hdl.handle.net/10803/284036.

[4] Nikolaos Petroulakis, Ioannis Askoxylakis, and Theo Tryfonas. Life-logging in smart environments: Challenges and security threats. In *International Conference on Communications (ICC)*, pages 5680–5684. IEEE, 2012.

[5] Paul Sarconi and Michael Calore. OK, house: get smart. How to Make the Most of Amazon Echo and Google Home. *Wired*, 25(6):39–41, 2017.

[6] M. Chau, X. Fang, and O. R. Liu Sheng. Analysis of the query logs of a web site search engine. *Journal of the American Society for Information Science and Technology*, 56:1363–1376, 2005.

[7] Peng Wang and Alan Smeaton. Using visual lifelogs to automatically characterize everyday activities. *Information Sciences*, 230:147–161, 2013.

[8] Justin M Grimes, Paul T Jaeger, and Jimmy Lin. Weathering the storm: The policy implications of cloud computing, 2009.

[9] Hal Varian. Economics of information technology. Cambridge University Press, 2001.

[10] Emily Steel. A web pioneer profiles users by name, 2010.

[11] A. Viejo, D. Sánchez, and J. Castellà-Roca. Preventing automatic user profiling in web 2.0 applications. *Knowledge-Based Systems*, 36:191–205, 2012.

[12] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Privacy protection in personalized search. *SIGIR Forum*, 41(1):4–17, 2007. doi: 10.1145/1273221.1273222. URL http://portal.acm.org/citation.cfm?id=1273222&dl=GUIDE&coll=GUIDE&CFID=17786915&CFTOKEN=37653058.

[13] Yabo Xu, Ke Wang, Benyu Zhang, and Zheng Chen. Privacy-enhancing personalized web search. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 591–600, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-654-7. doi: http://doi.acm.org/10.1145/1242572.1242652. URL http://portal.acm.org/citation.cfm?id=1242652.

[14] Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual ACM SIGIR conference*, pages 19–26, Seattle, Washington, USA, 2006. ACM. ISBN 1-59593-369-7. doi: 10.1145/1148170.1148177. URL http://portal.acm.org/citation.cfm?id=1148170.1148177.

[15] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 387–396, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. doi: http://doi.acm.org/10.1145/1135777.1135835. URL http://portal.acm.org/citation.cfm?id=1135777.1135835.

[16] Alissa Cooper. A survey of query log privacy-enhancing techniques from a policy perspective. *ACM Trans. Web*, 2(4):19:1–19:27, October 2008. ISSN 1559-1131. doi: 10.1145/1409220.1409222. URL http://doi.acm.org/10.1145/1409220.1409222.

[17] Rob Turtle Scott Cameron, Andrew Pickersgill. New ways for turning data into dollars now, 2014. https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/new-ways-for-turning-data-into-dollars-now.

[18] David J. Brenes and Daniel Gayo-Avello. Stratified analysis of aol query log. *Inf. Sci.*, 179:1844–1858, May 2009. ISSN 0020-0255. doi: 10.1016/j.ins.2009.01.027. URL http://portal.acm.org/citation.cfm?id=1523512.1523572.

[19] Barbara Poblete, Myra Spiliopoulou, and Ricardo A. Baeza-Yates. Website privacy preservation for query log publishing. In Francesco Bonchi, Elena Ferrari, Bradley Malin, and Yücel Saygin, editors, *Proc. of the 1st ACM SIGKDD international conference on Privacy, security, and trust in KDD – PinKDD'07*, volume 4890 of *Lecture Notes in Computer Science*, pages 80–96. Springer, 2007. ISBN 978-3-540-78477-7. URL https://link.springer.com/chapter/10.1007%2F978-3-540-78478-4_5.

[20] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl, editors, *Proc. of the 18th International World Wide Web Conference – WWW '09*, pages 171–180. ACM, 2009. ISBN 978-1-60558-487-4. URL https://dl.acm.org/citation.cfm?doid=1526709.1526733.

[21] B. J. Jansen. Search log analysis: What is it; what's been done; how to do it. *Library and Information Science Research*, 28(3):407–432, 2006.

[22] M. Richardson. Learning about the world through long-term query logs. *ACM Transactions on the Web*, 2(4), 2008.

[23] B. Tancer. *Click: What Millions of People Are Doing Online and Why it Matters*. Hyperion, 2008.

[24] Judit Bar-Ilan. Access to Query Logs — An Academic Researcher's Point of View. In Einat Amitay, G. Craig Murray, and Jaime Teevan, editors, *Query Log Analysis: Social And Technological Challenges. A workshop at the 16th International World Wide Web Conference (WWW 2007)*, May 2007.

[25] Peter Shea. Book review: 'click: What millions of people are doing online and why it matters' by bill tancer. *eLearn Magazine*, 2010(1):8, 2010. URL http://dblp.uni-trier.de/db/journals/elearn/elearn2010.html#Shea10.

[26] E. Adar. User 4xxxxx9: Anonymizing query logs. In *Workshop on Query Log Analysis at the 16th World Wide Web Conference*, 2007.

[27] Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, David Grossman, and Ophir Frieder. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 321–328, New York, NY, USA, 2004. ACM. ISBN 1-58113-881-4. doi: 10.1145/1008992.1009048. URL http://doi.acm.org/10.1145/1008992.1009048.

[28] Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, Ophir Frieder, and David Grossman. Temporal analysis of a very large topically categorized web query log. *J. Am. Soc. Inf. Sci. Technol.*, 58(2):166–178, January 2007. ISSN 1532-2882. doi: 10.1002/asi.v58:2. URL http://dx.doi.org/10.1002/asi.v58:2.

[29] Arnau Erola, Jordi Castellà-Roca, Guillermo Navarro-Arribas, and Vicenç Torra. Semantic microaggregation for the anonymization of query logs using the open directory project. *SORT - Statistics and Operations Research Transactions*, pages 41–58, 2011.

[30] Fabrizio Silvestri. Mining query logs: Turning search usage data into knowledge. *Found. Trends Inf. Retr.*, 4(1—2):1–174, January 2010. ISSN

1554-0669. doi: 10.1561/1500000013. URL http://dx.doi.org/10.1561/1500000013.

[31] K. Saagar. Monetizing data: Milking the new cash cow. *Wired Insights*, 2014.

[32] Andrew Pickersgill Scott Cameron and Rob Turtle. New ways for turning data into dollars now, 2014. http://www.forbes.com/sites/mckinsey/2014/01/15/new-ways-for-turning-data-into-dollars-now/.

[33] Leon Willenborg and Ton De Waal. *Elements of statistical disclosure control*, volume 155. Springer Science & Business Media, 2012.

[34] Christopher Soghoian. The problem of anonymous vanity searches. *ISJLP*, 3:299, 2007.

[35] U.S. Federal Trade Commission. Data brokers, a call for transparency and accountability, 2014.

[36] George Danezis, Josep Domingo-Ferrer, Marit Hansen, Jaap-Henk Hoepman, Daniel Le Metayer, Rodica Tirtea, and Stefan Schiffner. Privacy and data protection by design-from policy to engineering. *arXiv preprint arXiv:1501.03726*, 2015.

[37] Regulation(USA). Privacy Act of 1974, 1974. https://www.govinfo.gov/content/pkg/STATUTE-88/pdf/STATUTE-88-Pg1896.pdf.

[38] Council Europe. Proposal for a regulation of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data. In *General Data Protection Regulation*, January 2016, 2016.

[39] R. Jones, R. Kumar, B. Pang, and A. Tomkins. "i know what you did last summer": query logs and user privacy. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 909–914, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9. doi: http://doi.acm.org/10.

1145/1321440.1321573. URL http://portal.acm.org/citation.cfm?id=1321440.1321573&coll=GUIDE&dl=GUIDE.

[40] Michael Barbaro, Tom Zeller, and Saul Hansell. A face is exposed for aol searcher no. 4417749. *New York Times*, 9(2008), 2006. URL https://www.nytimes.com/2006/08/09/technology/09aol.html.

[41] Electronic Frontier Foundation. Aol's massive data leak, 2009. http://w2.eff.org/Privacy/AOL/.

[42] Elinor Mills. Aol sued over web search data release, September 2006. URL https://www.cnet.com/news/aol-sued-over-web-search-data-release/.

[43] S. Hansell. Increasingly, internet's data trail leads to court, 2006.

[44] Anton Nijholt. Google Home: Experience, support and re-experience of social home activities. *Information Sciences*, 178(3):612–630, 2008.

[45] David Pàmies-Estrems, Jordi Castellà-Roca, and Alexandre Viejo. Working at the Web Search Engine Side to Generate Privacy-preserving User Profiles. *Expert Systems with Applications*, 64(C):523–535, December 2016. ISSN 0957-4174. doi: 10.1016/j.eswa.2016.08.033. URL https://doi.org/10.1016/j.eswa.2016.08.033.

[46] David Pàmies-Estrems, Nesrine Kaaniche, Maryline Laurent, Jordi Castella-Roca, and Joaquin Garcia-Alfaro. Lifelogging Protection Scheme for Internet-based Personal Assistants. In Joaquin Garcia-Alfaro, Jordi Herrera-Joancomartí, Giovanni Livraga, and Ruben Rios, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology (CBT 2018 and DPM 2018)*, pages 431–440, Cham, September 2018. Springer. ISBN 978-3-030-00305-0. doi: 10.1007/978-3-030-00305-0_31. URL https://doi.org/10.1007/978-3-030-00305-0_31.

[47] David Pàmies-Estrems, Jordi Castellà-Roca, and Joaquin Garcia-Alfaro. A Real-Time Query Log Protection Method for Web Search Engines. *IEEE*

Access, May 2020. doi: 10.1109/ACCESS.2020.2992012. URL https://doi.org/10.1109/ACCESS.2020.2992012.

[48] Claudio Bettini and Daniele Riboni. Privacy protection in pervasive systems: State of the art and technical challenges. *Pervasive and Mobile Computing*, 17:159–174, 2015.

[49] Wikipedia. AOL search data leak. http://bit.ly/2JIGOqg, 2009.

[50] Cristina Romero-Tris, Alexandre Viejo, and Jordi Castellà-Roca. Multi-party methods for privacy-preserving web search: Survey and contributions. In *Advanced Research in Data Privacy*, pages 367–387. Springer, 2015. doi: 10.1007/978-3-319-09885-2_20. URL https://doi.org/10.1007/978-3-319-09885-2_20.

[51] A. Erola. *Contributions to privacy in Web Search Engines*. Universitat Rovira i Virgili, 2013. URL http://hdl.handle.net/10803/130934.

[52] Ask.com. Ask.com puts you in control of your search privacy with the launch of 'askeraser', 2007. http://www.irconnect.com/ask/pages/news_releases.html?d=132847.

[53] Center for Democracy and Technology. Search privacy practices: A work in progress, 2007. https://cdt.org/wp-content/uploads/privacy/20070808searchprivacy.pdf.

[54] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International, 1998.

[55] Yeye He and Jeffrey F. Naughton. Anonymization of set-valued data via top-down, local generalization. *Proc. VLDB Endow.*, 2(1):934–945, August 2009. ISSN 2150-8097. doi: 10.14778/1687627.1687733. URL http://dx.doi.org/10.14778/1687627.1687733.

[56] Manolis Terrovitis, Nikos Mamoulis, and Panos Kalnis. Privacy-preserving anonymization of set-valued data. *Proc. VLDB Endow.*, 1(1):115–125,

August 2008. ISSN 2150-8097. doi: 10.14778/1453856.1453874. URL http://dx.doi.org/10.14778/1453856.1453874.

[57] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL http://doi.acm.org/10.1145/219717.219748.

[58] Sabrina De Capitani di Vimercati, Sara Foresti, Giovanni Livraga, and Pierangela Samarati. Foundations of security analysis and design vi. In Alessandro Aldini and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design VI*, chapter Protecting Privacy in Data Release, pages 1–34. Springer-Verlag, Berlin, Heidelberg, 2011. ISBN 978-3-642-23081-3. URL http://dl.acm.org/citation.cfm?id=2028200.2028202.

[59] Anco Hundepool, Josep Domingo-Ferrer, Luisa Franconi, Sarah Giessing, Rainer Lenz, Jane Longhurst, E Schulte Nordholt, Giovanni Seri, and P Wolf. Handbook on statistical disclosure control. *ESSnet on Statistical Disclosure Control*, 2010.

[60] Yuan Hong, Xiaoyun He, Jaideep Vaidya, Nabil Adam, and Vijayalakshmi Atluri. Effective anonymization of query logs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management – CIKM'09*, CIKM '09, pages 1465–1468, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-512-3. doi: 10.1145/1645953.1646146. URL http://doi.acm.org/10.1145/1645953.1646146.

[61] Guillermo Navarro-Arribas and Vicenç Torra. Tree-based microaggregation for the anonymization of search logs. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, WI-IAT '09, pages 155–158, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3801-3. doi: 10.1109/WI-IAT.2009.251. URL http://dx.doi.org/10.1109/WI-IAT.2009.251.

[62] David Sánchez, Jordi Castellà-Roca, and Alexandre Viejo. Knowledge-based scheme to create privacy-preserving but semantically-related queries for web search engines. *Information Sciences*, 218:17–30, January 2013. ISSN 0020-0255. doi: 10.1016/j.ins.2012.06.025. URL https://doi.org/10.1016/j.ins.2012.06.025.

[63] Arnau Erola and Jordi Castellà-Roca. Using search results to microaggregate query logs semantically. In *Data Privacy Management and Autonomous Spontaneous Security - 8th International Workshop, DPM 2013, and 6th International Workshop, SETOP 2013, Egham, UK, September 12-13, 2013, Revised Selected Papers*, pages 148–161, 2013. doi: 10.1007/978-3-642-54568-9_10. URL https://doi.org/10.1007/978-3-642-54568-9_10.

[64] Montserrat Batet, Arnau Erola, David Sánchez, and Jordi Castellà-Roca. Utility preserving query log anonymization via semantic microaggregation. *Inf. Sci.*, 242:49–63, 2013. doi: 10.1016/j.ins.2013.04.020. URL https://doi.org/10.1016/j.ins.2013.04.020.

[65] Montserrat Batet, Arnau Erola, David Sánchez, and Jordi Castellà-Roca. Semantic anonymisation of set-valued data. In *ICAART 2014 - Proceedings of the 6th International Conference on Agents and Artificial Intelligence, Volume 1, ESEO, Angers, Loire Valley, France, 6-8 March, 2014*, pages 102–112, 2014. doi: 10.5220/0004811901020112. URL https://doi.org/10.5220/0004811901020112.

[66] Guillermo Navarro-Arribas, Vicenç Torra, Arnau Erola, and Jordi Castellà-Roca. User k-anonymity for privacy preserving data mining of query logs. *Inf. Process. Manage.*, 48(3):476–487, 2012. doi: 10.1016/j.ipm.2011.01.004. URL https://doi.org/10.1016/j.ipm.2011.01.004.

[67] Alexandre Viejo, David Sánchez, and Jordi Castellà-Roca. Preventing automatic user profiling in web 2.0 applications. *Knowl.-Based Syst.*, 36:191–205, 2012. doi: 10.1016/j.knosys.2012.07.001. URL https://doi.org/10.1016/j.knosys.2012.07.001.

[68] Claudio Carpineto and Giovanni Romano. Semantic search log k-anonymization with generalized k-cores of query concept graph. In *Advances in Information Retrieval - 35th European Conference on IR Research, ECIR 2013, Moscow, Russia, March 24-27, 2013. Proceedings*, pages 110–121, 2013. doi: 10.1007/978-3-642-36973-5_10. URL https://doi.org/10.1007/978-3-642-36973-5_10.

[69] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 1–12, 2006. doi: 10.1007/11787006_1. URL https://doi.org/10.1007/11787006_1.

[70] Palanivel A. Kodeswaran and Evelyne Viegas. Applying differential privacy to search queries in a policy based interactive framework. In *Proceeding of the ACM First International Workshop on Privacy and Anonymity for Very Large Databases, CIKM-PAVLAD 2009, Hong Kong, China, November 6, 2009*, pages 25–32, 2009. doi: 10.1145/1651449.1651455. URL https://doi.org/10.1145/1651449.1651455.

[71] Xuying Meng, Zhiwei Xu, Bo Chen, and Yujun Zhang. Privacy-preserving query log sharing based on prior n-word aggregation. In *2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, August 23-26, 2016*, pages 722–729, 2016. doi: 10.1109/TrustCom.2016.0131. URL https://doi.org/10.1109/TrustCom.2016.0131.

[72] Sicong Zhang, Hui Yang, and Lisa Singh. Applying epsilon-differential private query log releasing scheme to document retrieval. In *Proceedings of the 2nd International Workshop on Privacy-Preserving Information Retrieval Workshop PIR'15 2015, Santiago, Chile, August 13th, 2015*, 2015.

[73] Sicong Zhang, Grace Hui Yang, and Lisa Singh. Anonymizing query logs by differential privacy. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 753–756, 2016. doi: 10.1145/2911451.2914732. URL https://doi.org/10.1145/2911451.2914732.

[74] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: A review. *ACM SIGMOD Record*, 34(2):18–26, June 2005. ISSN 0163-5808. doi: 10.1145/1083784.1083789. URL http://doi.acm.org/10.1145/1083784.1083789.

[75] Georg Krempl, Indre Žliobaite, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, Sonja Sievi, Myra Spiliopoulou, and Jerzy Stefanowski. Open challenges for data stream mining research. *ACM SIGKDD Explorations Newsletter*, 16(1):1–10, September 2014. ISSN 1931-0145. doi: 10.1145/2674026.2674028. URL http://doi.acm.org/10.1145/2674026.2674028.

[76] R. Moore. Controlled data swapping techniques for masking public use microdata sets, 1996. (unpublished manuscript).

[77] Steven P. Reiss. Practical data-swapping: The first steps. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 14-16, 1980*, pages 38–45, 1980. doi: 10.1109/SP.1980.10014. URL https://doi.org/10.1109/SP.1980.10014.

[78] Vicenç Torra and Josep Domingo-Ferrer. *Disclosure control methods and information loss for microdata*, pages 91–110. Elsevier, 2001.

[79] Josep Domingo-Ferrer and Vicenç Torra. *A quantitative comparison of disclosure control methods for microdata*, pages 111–133. Elsevier, 2001.

[80] Guillermo Navarro-Arribas and Vicenç Torra. Rank swapping for stream data. In Vicenç Torra, Yasuo Narukawa, and Yasunori Endo, editors, *Proc. of the 11th International Conference on Modeling Decisions for Artificial Intelligence – MDAI'14*, volume 8825 of *Lecture Notes in Computer Science*, pages 217–226. Springer International Publishing, 2014. ISBN 978-3-319-12053-9. doi: 10.1007/978-3-319-12054-6_19. URL http://dx.doi.org/10.1007/978-3-319-12054-6_19.

[81] David Sánchez, Montserrat Batet, Alexandre Viejo, Mercedes Rodriguez-Garcia, and Jordi Castellà-Roca. A semantic-preserving differentially private

method for releasing query logs. *Inf. Sci.*, 460-461:223–237, 2018. doi: 10.1016/j.ins.2018.05.046. URL https://doi.org/10.1016/j.ins.2018.05.046.

[82] Charu C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 901–909, 2005. URL http://www.vldb.org/archives/website/2005/program/paper/fri/p901-aggarwal.pdf.

[83] Jianpei Zhang Gaoming Yang, Jing Yang and Yan Chu. Research on data streams publishing of privacy preserving. In *Proc. of the 2010 IEEE International Conference on Information Theory and Information Security – ICITIS'10*, pages 199 – 202. IEEE, 2010. ISBN 978-1-4244-6942-0. doi: 10.1109/ICITIS.2010.5688761.

[84] Jianzhong Li, Beng Chin Ooi, and Weiping Wang 0001. Anonymizing streaming data for privacy protection. In Gustavo Alonso, José A. Blakeley, and Arbee L. P. Chen, editors, *Proc. of the IEEE 24th International Conference on Data Engineering*, pages 1367–1369. IEEE, 2008. URL http://dblp.uni-trier.de/db/conf/icde/icde2008.html#LiOW08.

[85] J. Zhang, J. C. Yang, J. Zhang, and Y. Yuan. Kids: K-anonymization data stream base on sliding window. In *Proceedings of the 2010 2nd International Conference on Future Computer and Communication*, pages V2–311–V2–316, Shanghai, China, 2010.

[86] Bin Zhou, Yi Han, Jian Pei, Bin Jiang, Yufei Tao, and Yan Jia. Continuous privacy preserving publishing of data streams. In *Proc. of the 12th International Conference on Extending Database Technology: Advances in Database Technology – EDBT'09*, EDBT '09, pages 648–659, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-422-5. doi: 10.1145/1516360.1516435. URL http://doi.acm.org/10.1145/1516360.1516435.

[87] Kun Guo and Qishan Zhang. Fast clustering-based anonymization approaches with time constraints for data streams. *Knowledge-Based Systems*, 46:95–108, July 2013. ISSN 0950-7051. doi: 10.1016/j.knosys.2013.03.007. URL http://dx.doi.org/10.1016/j.knosys.2013.03.007.

[88] Hessam Zakerzadeh and Sylvia L. Osborn. Faanst: Fast anonymizing algorithm for numerical streaming data. In Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, Ana Cavalli, and Jean Leneutre, editors, *Proc. of the 5th international Workshop on data privacy management – DPM'10*, volume 6514 of *Lecture Notes in Computer Science*, pages 36–50. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-19347-7. doi: 10.1007/978-3-642-19348-4_4. URL http://dx.doi.org/10.1007/978-3-642-19348-4_4.

[89] R. Mortazavi and S. Jalili. Fast data-oriented microaggregation algorithm for large numerical datasets. *Knowledge-Based Systems*, 67:195—205, 2014.

[90] Soohyung Kim, Min Kyoung Sung, and Yon Dohn Chung. A framework to preserve the privacy of electronic health data streams. *Journal of Biomedical Informatics*, 50:95–110, 2014.

[91] Jordi Soria-Comas and Josep Domingo-Ferrer. Probabilistic k-anonymity through microaggregation and data swapping. In *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, pages 1–8. IEEE, 2012.

[92] Julio Bondia-Barcelo, Jordi Castellà-Roca, and Alexandre Viejo. Building privacy-preserving search engine query logs for data monetization. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), Toulouse, France, July 18-21, 2016*, pages 390–397, 2016. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0074. URL https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0074.

[93] Helen F. Nissenbaum and Howe Daniel. Trackmenot: Resisting surveillance in web search. lessons from the identity trail: Anonymity, privacy, and identity in a networked society. *Oxford: Oxford University Press*, 2009. URL https://ssrn.com/abstract=2567412.

[94] J. Domingo-Ferrer, A. Solanas, and J. Castellà-Roca. h(k)-private information retrieval from privacy-uncooperative queryable databases. *Journal of Online Information Review*, 33(4):1468–1527, 2009.

[95] Mummoorthy Murugesan and Chris Clifton. Providing privacy through plausibly deniable search. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 768–779, 2009. doi: 10.1137/1.9781611972795.66. URL https://doi.org/10.1137/1.9781611972795.66.

[96] Avi Arampatzis, PavlosS. Efraimidis, and George Drosatos. A query scrambler for search privacy on the internet. *Information Retrieval*, pages 1–23, 2012. ISSN 1386-4564. doi: 10.1007/s10791-012-9212-1. URL http://dx.doi.org/10.1007/s10791-012-9212-1.

[97] Ero Balsa, Carmela Troncoso, and Claudia Díaz. OB-PWS: obfuscation-based private web search. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 491–505, 2012. doi: 10.1109/SP.2012.36. URL https://doi.org/10.1109/SP.2012.36.

[98] Panagiotis Papadopoulos, Antonis Papadogiannakis, Michalis Polychronakis, Apostolis Zarras, Thorsten Holz, and Evangelos P. Markatos. k-subscription: privacy-preserving microblogging browsing through obfuscation. In *Annual Computer Security Applications Conference, ACSAC '13, New Orleans, LA, USA, December 9-13, 2013*, pages 49–58, 2013. doi: 10.1145/2523649.2523671. URL https://doi.org/10.1145/2523649.2523671.

[99] Albin Petit, Thomas Cerqueus, Sonia Ben Mokhtar, Lionel Brunie, and Harald Kosch. PEAS: private, efficient and accurate web search. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20-22, 2015, Volume 1*, pages 571–580, 2015. doi: 10.1109/Trustcom.2015.421. URL https://doi.org/10.1109/Trustcom.2015.421.

[100] R. Chow and P. Golle. Faking contextual data for fun, profit, and privacy. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society – WPES'09*, pages 105–108, 2009.

[101] S. T. Peddinti and N. Saxena. On the privacy of web search based on query obfuscation: a case study of trackmenot. In *Proceedings of the 10th international conference on Privacy enhancing technologies – PETS'10*, pages 19–37, 2010.

[102] R. Al-Rfou', W. Jannen, and N. Patwardhan. TrackMeNot-so-good-after-all, November 2012. arXiv :211.0320.

[103] M. Reiter and A. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[104] Jordi Castellà-Roca, Alexandre Viejo, and Jordi Herrera-Joancomartí. Preserving user's privacy in web search engines. *Computer Communications*, 32 (13-14):1541–1551, 2009. doi: 10.1016/j.comcom.2009.05.009. URL https://doi.org/10.1016/j.comcom.2009.05.009.

[105] Y. Lindell and E. Waisbard. Private web search with malicious adversaries. In *Proceedings of the 10th international conference on Privacy enhancing technologies – PETS'10*, pages 220–235, 2010.

[106] A. Viejo and J. Castellà-Roca. Using social networks to distort users' profiles generated by web search engines. *Computer Networks*, 54(9):1343–1357, 2010.

[107] A. Erola, J. Castellà-Roca, A. Viejo, and J. M. Mateo-Sanz. Exploiting social networks to provide privacy in personalized web search. *Journal of Systems and Software*, 84(9):1734–1745, 2011.

[108] C. Romero-Tris, A. Viejo, and J. Castellà-Roca. Improving query delay in private web search. In *3PGCIC*, pages 200–206, 2011.

[109] C. Romero-Tris, J. Castellà-Roca, and A. Viejo. Multi-party private web search with untrusted partners. In *7th International ICST Conference on Security and Privacy in Communication Networks –SecureComm'11*, 2011.

[110] George Danezis and Claudia Diaz. A survey of anonymous communication channels. Technical Report MSR-TR-2008-35, Microsoft, February 2008. URL https://www.microsoft.com/en-us/research/publication/a-survey-of-anonymous-communication-channels/.

[111] G. Danezis, C. Diaz, and P. Syverson. Systems for anonymous communication. In *CRC cryptography and network security series*. Chapman Hall/CRC, 2009.

[112] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981. ISSN 0001-0782. doi: 10.1145/358549.358563. URL http://doi.acm.org/10.1145/358549.358563.

[113] Brian Neil Levine and Clay Shields. Hordes: A multicast based protocol for anonymity. *Journal of Computer Security*, 10:213–240, 2002.

[114] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web mixes: a system for anonymous and unobservable internet access. In *International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability*, pages 115–129, New York, NY, USA, 2001. Springer-Verlag New York, Inc. ISBN 3-540-41724-9. URL http://dl.acm.org/citation.cfm?id=371931.371983.

[115] Benedikt Westermann, Rolf Wendolsky, Lexi Pimenidis, and Dogan Kesdogan. Cryptographic protocol analysis of an.on. In *Proceedings of the 14th international conference on Financial Cryptography and Data Security*, FC'10, pages 114–128, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-14576-0, 978-3-642-14576-6. doi: 10.1007/978-3-642-14577-3_11. URL http://dx.doi.org/10.1007/978-3-642-14577-3_11.

[116] Rainer Bohme, George Danezis, Claudia Diaz, Stefan Kapsell, and Andreas Pfitzmann. Mix cascades vs. peer-to-peer: Is one concept superior? In *In Privacy Enhancing Technologies (PET 2004)*, 2004.

[117] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42:39–41, 1999.

[118] F. Saint-Jean, A. Johnson, D. Boneh, and J. Feigenbaum. Private web search. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society – WPES'07*, pages 84–90, 2007.

[119] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1251375.1251396.

[120] Felipe Astolfi, Jelger Kroese, and Jeroen Van Oorschot. I2p - the invisible internet project. Web technology report, Media Technology, Leiden University, 2015.

[121] Paul Syverson. Practical vulnerabilities of the tor anonymity network. In *Advances in Cyber Security: Technology, Operation and Experiences*, 2011.

[122] Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords, 1997.

[123] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 364–373. IEEE press, 1997.

[124] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, 1998.

[125] R. Ostrovsky and W. E. Skeith-III. A survey of single-database pir: techniques and applications. In *Lecture Notes in Computer Science*, volume 4450, pages 393–411, 2007.

[126] Lorrie Faith Cranor, Praveen Guduru, and Manjula Arjula. User interfaces for privacy agents. *ACM Transactions Computer-Human Interaction*, 13(2): 135–178, June 2006. ISSN 1073-0516. doi: 10.1145/1165734.1165735. URL http://doi.acm.org/10.1145/1165734.1165735.

[127] Lorrie Faith Cranor, Serge Egelman, Steve Sheng, Aleecia M. McDonald, and Abdur Chowdhury. P3p deployment on websites. *Electronic Commerce Research and Applications*, 7(3):274–293, 2008.

[128] C. Soghoian. The history of the do not track header. Slight Paranoia, February 2012.

[129] J Mayer, A Narayanan, and S Stamm. Do not track: A universal third-party web tracking opt out. *IETF Request for Comments*, pages 1–12, 2011.

[130] Martin Beck and Michael Marhfer. Do-not-track techniques for browsers and their implications for consumers. In Jan Camenisch, Bruno Crispo, Simone Fischer-Hübner, Ronald Leenes, and Giovanni Russello, editors, *Privacy and Identity Management for Life*, volume 375 of *IFIP Advances in Information and Communication Technology*, pages 187–196. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-31667-8. doi: 10.1007/978-3-642-31668-5_14. URL http://dx.doi.org/10.1007/978-3-642-31668-514.

[131] Omer Tene and Jules Polenetsky. To track or do not track: Advancing transparency and individual control in online behavioral advertising. *Minn. JL Sci. & Tech.*, 13:281, 2012.

[132] Harry Hochheiser. The platform for privacy preference as a social protocol: An examination within the u.s. policy context. *ACM Transactions Internet Technology*, 2(4):276–306, November 2002. ISSN 1533-5399. doi: 10.1145/604596.604598. URL http://doi.acm.org/10.1145/604596.604598.

[133] Ian Reay, Scott Dick, and James Miller. A large-scale empirical study of p3p privacy policies: Stated actions vs. legal obligations. *ACM Transactions Web*, 3(2):6:1–6:34, April 2009. ISSN 1559-1131. doi: 10.1145/1513876.1513878. URL http://doi.acm.org/10.1145/1513876.1513878.

[134] Electronic Privacy Information Center (EPIC). Pretty poor privacy: An assessment of p3p and internet privacy. http://epic.org/reports/prettypoorprivacy.html, June 2000.

[135] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Ucair: Capturing and exploiting context for personalized search. In *Proceedings of the ACM SIGIR 2005 Workshop on Information Retrieval in Context (IRiX)*, page 45. Citeseer, 2005.

[136] Kenneth Wai-Ting Leung, Dik Lun Lee, Wilfred Ng, and Hing Yuet Fung. A framework for personalizing web search with concept-based user profiles. *ACM Transactions Internet Technology*, 11(4):17:1–17:29, March 2012. ISSN 1533-5399. doi: 10.1145/2109211.2109214. URL http://doi.acm.org/10.1145/2109211.2109214.

[137] Tomáš Kramár, Michal Barla, and Mária Bieliková. Personalizing search using socially enhanced interest model, built from the stream of user's activity. *Journal Web Engineering*, 12(1-2):65–92, February 2013. ISSN 1540-9589. URL http://dl.acm.org/citation.cfm?id=2481562.2481565.

[138] J. Soria-Comas and J. Domingo-Ferrer. Big data privacy: challenges to privacy principles and models. *Data Science and Engineering*, 1(1):1–8, 2015.

[139] M. Batet, A. Erola-Ferrer, D. Sánchez, and J. Castellà-Roca. Utility preserving query log anonymization via semantic microaggregation. *Information Sciences*, 242(1):49—63, 2013.

[140] Netcraft. January 2020 Web Server Survey, 2020. https://news.netcraft.com/archives/2020/01/21/january-2020-web-server-survey.html.

[141] R. Jones, R. Kumar, B. Pang, and A. Tomkins. I know what you did last summer. In *Proc. of the 16th ACM conference on Conference on information and knowledge management – CIKM'07*, 2007.

[142] V. Torra and J. Domingo-Ferrer. Disclosure control methods and information loss for microdata. In *Confidentiality, disclosure, and data access: Theory and practical applications for statistical agencies*. Elsevier, 2001.

[143] A. B. Bondi. Characteristics of scalability and their impact on performance. In *Proc. of the 2nd International Workshop on Software and Performance*, 2000.

[144] M. Michael, J. E. Moreira, D. Shiloach, and R. W. Wisniewski. Scale-up x scale-out: A case study using nutch/lucene. In *Proc. of the 2007 IEEE International Parallel & Distributed Processing Symposium –IPDPS'07*, 2007.

[145] Internet Live Stats. Google search statistics. http://www.internetlivestats.com, 2016.

[146] U.S. Department of Health and Human Services. Health insurance portability and accountability act. http://www.hhs.gov/hipaa, 1996.

[147] D. Sánchez, M. Batet, A. Valls, and K. Gibert. Ontology-driven web-based semantic similarity. *Journal of Intelligent Information Systems*, 35(3):383–413, 2010.

[148] A. Viejo and D. Sánchez. Profiling social networks to provide useful and privacy-preserving web searchs. *Journal of the Association for Information Science and Technology*, 65(12):2444–2458, 2014.

[149] André B Bondi. Characteristics of scalability and their impact on performance. In *Proceedings of the 2nd international workshop on Software and performance*, pages 195–203. ACM, 2000.

[150] Maged Michael, Jose E Moreira, Doron Shiloach, and Robert W Wisniewski. Scale-up x scale-out: A case study using nutch/lucene. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8. IEEE, 2007.

[151] Inc. AOL. Aol keyword searches., 2006. http://dontdelete.com/default.asp.

[152] Kingsley Purdam and Mark Elliot. A case study of the impact of statistical disclosure control on data quality in the individual uk samples of anonymised records. *Environment and Planning A*, 39(5):1101–1118, 2007.

[153] A. Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *G. Ist. Ital. Attuari*, 4:83–91, 1933.

[154] N. Smirnov. Table for estimating the goodness of fit of empirical distributions. *The annals of mathematical statistics*, 19(2):279–281, 1948.

[155] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[156] Internet Live Stats. Google search statistics, 2017. http://www.internetlivestats.com/google-search-statistics/.

[157] César Gutiérrez, Juan Garbajosa, Jessica Diaz, and Agustin Yagüe. Providing a Consensus Definition for the Term Smart Product. In *20th International Conference and Workshops on the Engineering of Computer Based Systems*, pages 203–211, 2013.

[158] European Parliament and Council of the European Union. Directive 95/46/ec of the european parliament and of the council, 1995. http://bit.ly/2JGKSaB.

[159] Netcraft. January 2018 Web Server Survey, 2018. http://bit.ly/2JTEoRL.

[160] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable signatures. In *10th European Conference on Research in Computer Security*, ESORICS'05, pages 159–177, Berlin, Heidelberg, 2005. Springer-Verlag.

[161] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schroder. Unlinkability of sanitizable signatures. In *13th International Conference on Practice and Theory in Public Key Cryptography*, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-13012-7, 978-3-642-13012-0.

[162] Sébastien Canard and Amandine Jambert. On extended sanitizable signature schemes. In *2010 International Conference on Topics in Cryptology*, CT-RSA'10, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-11924-7, 978-3-642-11924-8.

[163] Pravin Shankar, Vinod Ganapathy, and Liviu Iftode. Privately querying location-based services with SybilQuery. In *11th international conference on Ubiquitous computing*, pages 31–40. ACM, 2009.

[164] Mohamed Mokbel, Chi-Yin Chow, and Walid Aref. The new casper: Query processing for location services without compromising privacy. In *32nd international conference on Very large data bases*, pages 763–774, 2006.

[165] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *International conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

[166] Weiran Liu. A library for cryptographic primitive implementations for cloud storage applications., 2017. https://github.com/liuweiran900217/CloudCrypto.

[167] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[168] Claudio De Stefano, Carlo Sansone, and Mario Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(1):84–94, 2000.

UNIVERSITAT
ROVIRA i VIRGILI