A REVIEW ON THE SECURITY AND USABILITY OF DIFFERENTIAL PRIVACY

JUAN JOSÉ MATA DE ACUÑA

Submitted in partial fulfilment of the requirements of Napier University for the degree of Master of Science in Advanced Security and Digital Forensics

Edinburgh Napier University

School of Computing

August 2017

Authorship Declaration

I, Juan José Mata de Acuña, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained informed consent from all people I have involved in the work in this dissertation following the School's ethical guidelines.

Signed:

Date:

Matriculation no: 40291916

Data Protection Declaration

Under the 1998 Data Protection Act, The University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please write your name below one of the options below to state your preference.

☐ The University may make this dissertation, with indicative grade, available to others.

☐ The University may make this dissertation available to others, but the grade may not be disclosed.

☐ The University may not make this dissertation available to others.

Abstract

The increasing trend for data collection in modern companies creates a challenge for privacy protection. Cases where the advances in data processing have been used to recover identifiable information from published, curated datasets have started to disprove traditional methods of anonymisation. This, together with the progressively tightening legal framework being expanded with regulations like the General Data Protection Regulation (GDPR), which ultimate aim is to protect the privacy of users, creates an urgent necessity for developing more thorough and verifiable privacy preservation mechanisms.

In response to this problem, different lines of research have risen, with varied propositions that seek to tackle this issue. Techniques like randomisation, anonymisation, query auditing or generalisation have been proposed by academic researchers to protect the privacy of the users. Inside the randomisation category, a promising concept called differential privacy can be highlighted. This approach proposes a formal definition of privacy, based on solid mathematical foundations and supported by companies like Google or Microsoft in the late years.

This work aims to contribute to the further study and development of techniques based on differential privacy. Using an open sourced project called Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR), created by Google researchers and incorporated into the Chrome web browser, a platform for generating RAPPOR simulations from a given dataset is produced to elaborate scenarios with different parameters that can be compared to evaluate the effects of altering the input values on the recovered data.

The experimental results show how the increase of the privacy requirement of the mechanism leads to producing noisier results, generating distributions increasingly distant to the original data as the privacy guarantee becomes more constraining. Also, it is observed how smaller populations suffer the effect of high privacy requirements more than populations with a higher number of reports. Finally, an additional experiment performed at the end of the project allowed to appreciate how this loss of fidelity in the recovered data grows linearly when the privacy guarantee is increased.

Keywords: privacy, anonymity, differential privacy, RAPPOR, GDPR, compliance

Contents

List of Figures

List of Tables

Acknowledgements

Chapter 1

Introduction

1.1 Background

The business community has become increasingly dependant of technology in recent years. The number of companies that significantly rely on the collection, study and transfer of user data for their economic activities continues growing at a fast pace (Narayanan & Shmatikov, 2010). From developing more customised experiences to detecting anomalies in the population that can indicate new malware campaigns (Erlingsson, Pihur & Korolova, 2014), information obtained from users is more and more a requisite for both private and public sector.

Transactions of datasets are performed continuously, finding examples like public health institutions releasing data with the intention of fostering medical research (El Emam, Jonker, Arbuckle & Malin, 2011), or big companies like Facebook, Google or Microsoft collecting all sorts of user data, selling them later to advertising companies and other partners that in some cases (Apple) are not even explicitly declared (Westrick, 2016; Morran, 2016; Hachman, 2015; Kosner, 2013).

Traditionally, data collectors have ensured a certain level of privacy preservation over released datasets through anonymisation and de-identification techniques. Unfortunately, recent studies and news indicate the insufficiency of this method: Cases as the re-identification of AOL users from released, anonymised search queries (Barbaro & Zeller, 2006); the identification of individuals via correlation of common demographic data like zip codes, gender and date of birth (Sweeney, 2000); or the work of Narayanan and Shmatikov (2006, 2008) on de-anonymising sanitised data from the "Netflix Prize data mining contest" demonstrate how the umbrella of Personal Identifiable Information (PII) becomes wider over time as new data association techniques (like machine learning and data mining) evolve.

Moreover, the legal environment is gradually tightening in regard to data protection. New regulations like the General Data Protection Regulation (GDPR) widens the scope of PII, increases the responsibility held by data processors

and controllers and stiffens sanctions imposed in case of data breaches or other incidents of personal data disclosure (Kelly, 2016). The requirements and limitations imposed by this new regulation are often vague and merely descriptive, creating an atmosphere of confusion among businesses.

Consequently, dilemmas and heated debates arise while discussing what data to collect and how to preserve the privacy of data donors. Selectively stripping sensitive information is inefficient, as it has been proven that any piece of data that differentiates any user from another can be employed to identify them (Narayanan & Shmatikov, 2010). Current lines of work put their efforts in developing higher and more formal guarantees around data that will be published eventually. In the middle of this ambient *differential privacy* appears. This approach proposes a new, formal definition of privacy, focussing on the study of population statistics while enforcing a strong deniability of individuals' data via the addition of noise (Dwork, Mcsherry, Nissim & Smith, 2006). Being the only privacy guarantee that offers objective metrics about the level of confidentiality provided by a method (Rossi, 2016), the theoretical work started by Dwork et al. has increasingly been incorporated to new research and practical cases (Wang, Wu & Hu, 2016; Giakkoupis, Guerraoui, Jégou, Kermarrec & Mittal, 2015; Hu et al., 2015), with examples of its integration in companies like Apple (whose senior vice president of software engineering described it as "future proof" (Greenberg, 2016)), Microsoft (McSherry, 2009) or Google (Erlingsson, 2014).

1.2 Aim and Objectives

This work seeks to contribute to the critical evaluation of differential privacy as a solid option for privacy preservation standards in the field of data collection. As it will be seen in the literature review, its wide acceptance by the academic community, together with the absence of solid arguments that disprove it, ensures a high level of privacy protection when a mechanism based on it is properly configured. Its main problem seems to be the lack of documentation or guidance to that suitable adjustment of parameters.

This thesis' main aim is to evaluate the usability of differential privacy and to establish some configuration guidelines that satisfy both privacy guarantees and data controllers' necessities. In order to accomplish this, the following objectives will be sought:

1. Critical literature review of current research in the field of privacy preservation techniques. Special attention will be paid to their advantages, limitations and real world implementations.
2. Design of a rigorous testing methodology that allows to compare the effects of altering parameters (e.g., population size, privacy guarantee, etc.) in the final reports obtained in different scenarios generated from the combination of those input parameters, with the intention of correlating these variations to characteristics like retrievability and privacy protection levels.
3. Implementation of a piece of software that enables conducting the experiment. For this part of the thesis, the Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR), an implementation of differential privacy incorporated into the Chrome browser, will be used. The artifact will consist on a pipeline of modules that will take the original data through the whole process until generating the RAPPOR reports for their later analysis.
4. Discussion and evaluation of the obtained results, comparing them to the original data and judging the relationships between the different input and output metrics.

Chapter 2

Literature Review

2.1 Introduction

With the intention of analysing the landscape of alternatives in the field of privacy preservation techniques, the following chapter offers a literature review that aims to critically reflect on the diverse methods available for this purpose.

First, a differentiation between privacy and anonymity is made, as the two terms will be often mentioned and should not be confused.

Second, a classification of different privacy preservation methods is made with the intention of providing a framework under which concrete techniques can be evaluated more objectively. Characteristics, limitations and real life examples will be depicted from each of the groups of mechanisms.

Third, a critical analysis of differential privacy, a formal privacy definition born inside the noise addition category, is made. This method has gained traction in the late years and is supported by a wide variety of academic researchers and strong companies.

Finally, the Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR) mechanism is shown as a specific implementation of the differential privacy mechanism, studying its qualities and limitations.

2.1.1 Privacy vs Anonymity

These concepts are commonly mistaken and even interchanged. As both of them will be mentioned extensively along this document, they must be correctly defined from the beginning:

Anonymity According to Pfitzmann and Köhntopp (2001), an individual will remain anonymous whenever they can perform an action without their identity being discovered. The definition of anonymity is usually a straightforward concept, regardless of the context.

Goddyn (2001) states that, in the electronic world, anonymity can be both beneficial and problematic: On the one hand, the safety of performing actions while remaining anonymous can encourage people to perform ethical actions that could be punished if their identity was known, as denouncing the violation of human rights in a totalitarian regime. On the other hand, it can be used by malicious actors to perform illegal activities that should be prosecuted, like attacking some company servers.

Privacy Even though it is recognised in The Universal Declaration of Human Rights, the definition of privacy is often dependant on the concrete legal definition of each country or context (data privacy, political, medical, etc.). Generally speaking, privacy can be understood as the right of an individual or group of individuals to keep information of any kind from third parties (Harkiolakis, 2013).

To set an example closer to the environment in which this work will develop, it could be said that an user will remain anonymous whenever their participation in a survey remains unknown, while their responses would be exposed and, therefore, no longer be private. On the other hand, knowing that this individual collaborated in the survey would damage their anonymity, while their privacy would remain unharmed as long as their answers remained unknown.

2.2 Data Collection and Privacy Preservation

In order to study tendencies about a population, data about its members has to be collected. Protecting the privacy of the users that provided such data is both an ethic and a legal responsibility for data controllers, and a wide variety of methods have been developed over the years to this end. In this section, these methods will be described and catalogued to offer a more complete view of the landscape where this work is developed.

2.2.1 Privacy Preservation Techniques

Interactive and Non-Interactive Approaches

Authors like Dwork et al. (2006) emphasise the importance of dividing the different techniques into interactive and non-interactive approaches. According to them, a non-interactive approach would imply not publishing the data itself, but a curated or stripped version of it (Dwork, 2011b). This would be the case

of anonymised or de-identified datasets. In the interactive approach, an interface is offered to access the original data through queries that will provide information after passing through some privacy filter, like query analysis or noise addition (Dwork & Roth, 2013, p. 11).

Non-Interactive Approach In the non-interactive approach, the data controller will no longer be responsible for the original data, as this one will be never published. The only risk of disclosure would remain over the released data, this being statistics about the collected information, or an anonymised or subsampled version of it (Dwork, 2011b). The main disadvantage of this method is that the utility of the data must be defined at the moment of the sanitisation, being highly difficult to expand it later on (Dwork, 2006). The destruction of the original data is a valid option according to this model, as long as no new information is planned to be released about it (Dwork, 2008).

Interactive Approach The interactive approach implies that the data controller must be continuously maintaining the interface for accessing the data, allowing an adaptable utility for researchers in return (Dwork, 2011b). Moreover, Dwork et al. (2006) demonstrate that any non-interactive sanitisation that satisfies differential privacy (which is extensively explained on Section 2.3) can be replaced for a set of interactive, low sensitivity queries that will offer more versatility to the utility of the data, as the sanitisation has to be designed to satisfy some functions, excluding others.

Classification Criteria

Most commonly, literature focuses on expanding or evaluating a concrete method instead of surveying and cataloguing existent works. Only a few papers (Aggarwal & Yu, 2008a; Dwork, 2011b; Article 29 Data Protection Working Party, 2014) offer a broad view of the available alternatives on privacy preservation techniques, and each author groups the methods according to their own criteria. In an attempt to summarise the available technologies at the time of writing this work, a classification is offered in the following lines. This classification has been strongly influenced by the work of Aggarwal and Yu (2008a), as this paper is the more focused one on the issue of classifying them. The paper published by the Article 29 Data Protection Working Party (2014) also addresses this, although they talk about the issue from the anonymity point of view. The methods discussed are essentially the same ones, but, as it has already been discussed, the goals of anonymity and privacy are considerably different.

Randomisation As its name suggests, the randomisation method consists of adding aleatory noise to the data to hide the real values of the records (Aggarwal & Yu, 2008b). Without further anonymisation operations, a single record can be still associated with an individual, but whatever the individual answered would be granted with a strong privacy protection (Article 29 Data Protection Working Party, 2014).

According to the definition of Aggarwal and Yu (2008a), randomisation would enter in the non-interactive approach, permanently distorting the data prior its storage. In practising this approach, the original data would not be recoverable, offering a safer but more limited option.

Opposed to the aforementioned, Dwork (2011b), who often refers to randomisation as noise addition, suggests an alternative, interactive approach to it: Differentiating between input and output perturbations, the original data would be stored in the database, adding the noise to the queries that ask for data about the population. In the case of input perturbation, the data would be altered before compiling the answer to the query, while in output perturbation the noise would be added to the answer created with the original data (Kasiviswanathan, Lee, Nissim, Raskhodnikova & Smith, 2011).

The mechanisms chosen to add noise must be designed carefully, as inconsistent noise may be filtered out by an attacker who would then be able to reconstruct the original records (Article 29 Data Protection Working Party, 2014). Differential privacy (Dwork et al., 2006; Dwork, 2006, 2011b) aims to solve this problem with a formal definition that imposes the amount and shape of noise that has to be added to satisfy a minimum threshold of privacy. This concept will be extensively covered in following sections. As the paper of Aggarwal and Yu (2008a) was published when differential privacy was in its early stages of development, only a swift mention of it is made on that work, and it appears in the query auditing section instead of in the randomisation section. This is not incorrect (original works on differential privacy emphasise its interactive properties and is always exemplified with perturbations applied to queries performed against the published database (Dwork et al., 2006; Dwork, 2006; McSherry, 2009; Dwork, 2011b)), but it is placed with the randomisation methods on later works (Article 29 Data Protection Working Party, 2014).

A particular form of adding noise is via the permutation of certain values in the records, shuffling the information between them. In this scenario, the overall results would be preserved, but the correlation between the data would be lost (Article 29 Data Protection Working Party, 2014). There is a high likelihood that permutation will not provide enough privacy by itself, as it is easy to reassociate shuffled values that possess a strong link between them. For instance, the ex-

ample shown by Article 29 Data Protection Working Party (2014) exhibits how it is possible to discover the salary of the employees of a company based on their position and date of birth (e.g., the income of the CEO will most likely be the highest one).

When retrieving useful information from altered data, a reconstruction of the statistics of the population has to be performed. It can be assumed that, with a certain level of confidence, truthful data about the population can be recovered through statistical tools as the Bayes reconstruction method or the Expectation-Maximisation method (Aggarwal & Yu, 2008a).

Generalisation The aim of this approach is to make it impossible for an adversary to associate a record in the database with an individual. This is performed by ensuring that multiple entries of the database will contain the same values for data that could be used to identify someone (Aggarwal & Yu, 2008a). Almost all the work made in this area is focused on $k-$anonymity (Sweeney, 2002), which formally defines a requirement by which all values or group of values that are not sensitive by themselves but can potentially be used to identify an individual (called quasi-identifiers) will be shared by at least k entries of the database for all cases. This is often done reducing the granularity of the attributes (e.g., collecting the country instead of the city) (Article 29 Data Protection Working Party, 2014).

The main problem with the initial proposition of $k-$anonymity is that it did not contemplate inference attacks, where the attacker can obtain information about a target from some additional source. The paper published by the Article 29 Data Protection Working Party (2014) shows an example of this, demonstrating how the medical condition of a patient can be learnt, assuming that the attacker knows their year of birth, as all the people in the dataset born in the same year had the same condition. More pieces of work that expose this design problem were already discussed in the Background section (Narayanan & Shmatikov, 2008, 2010)

An improved version of $k-$anonymity called $l-$diversity was presented by Machanavajjhala et al. in 2007. This theory aimed to solve the design flaws of $k-$anonymity by imposing a restriction by which a minimum of l different sensitive values should appear in a group of tuples that share the same quasi-identifiers.

As Table 2.1 (Machanavajjhala et al., 2007) shows, the quasi-identifier formed by the group of non-sensitive attributes, often called "equivalence class" (Li, Li & Venkatasubramanian, 2007), makes groups of 4 individuals ($k = 4$). This

table respects the $l-$diversity threshold of 3, as each group contains at least 3 different health conditions (the sensitive attribute).

Non-Sensitive			Sensitive
Zip Code	Age	Nationality	Condition
1305*	≤40	*	Heart Disease
1305*	≤40	*	Viral Infection
1305*	≤40	*	Cancer
1305*	≤40	*	Cancer
1305*	>40	*	Cancer
1305*	>40	*	Heart Disease
1305*	>40	*	Viral Infection
1305*	>40	*	Viral Infection
1305*	≥40	*	Heart Disease
1305*	≥40	*	Viral Infection
1305*	≥40	*	Cancer
1305*	≥40	*	Cancer

Table 2.1: 3-diverse database example (Machanavajjhala et al., 2007)

Short time after its first appearance, $l-$diversity was demonstrated to be inefficient. Li et al. (2007) show that the restriction it imposed was both unnecessary and insufficient. The authors proposed a new restriction called $t-$closeness, according to which the distribution of the sensible attribute of the equivalence class should be proportional to the distribution of such attribute in the entire dataset.

Nevertheless, not even this updated form of $k-$anonymity remains unscathed to critiques: According to Domingo-Ferrer and Torra (2008), the outcome of enforcing $t-$closeness is a dataset which usability is more than limited.

Query Auditing This approach to privacy preservation suggests the analysis of the queries issued against the database to determine if a query of group of queries can pose a data breach (Dwork, 2011b). Two different techniques are present in literature: If the auditing is done over queries already answered to determine whether a breach of private information already occurred, it is denominated offline auditing (Nabar, Kenthapadi, Mishra & Motwani, 2008). If, on the other hand, the incoming query is analysed to determine whether the answer to it, together with the answers already given for previous queries, incurs in a breach, it is called online auditing (Kenthapadi, Mishra & Nissim, 2005).

Query auditing seems to have lost traction in the last years, as not many papers have been published in this area in the last decade. Talking about offline auditing, this is probably due to the fact that it does not prevent the disclosure of data because of its a posteriori approach. Additionally, online auditing is often said to be computationally infeasible (Dwork, 2011b; Kleinberg, Papadimitriou & Raghavan, 2003), and the fact that blocking the answer to a query can be disclosive by itself (Kenthapadi et al., 2005) is not an incentive for its implementation either.

2.3 Differential Privacy

Differential privacy is a definition of privacy whose ultimate paradigm states that no new knowledge of any individual should be learnt from accessing the data (Dwork & Roth, 2013; Dwork, 2006). It's worth highlighting that differential privacy is a formal definition of a requirement that a privacy-preserving mechanism should satisfy, and not a method or algorithm by itself. Therefore, it can be said that a method is "differential privacy compliant", but it would be incorrect stating that any concrete mechanism represents differential privacy by itself.

The initial problem that it tried to solve was the necessity of acquiring information about population while protecting the privacy of the participants of the survey (Dwork, 2006). According to the differential privacy paradigm, the ability of an attacker to learn something about an individual should not not be related to the presence or absence of the person in the dataset. This is achieved thanks to the addition of noise to each entry of the database, which ensures that reliable data will only be extracted from big groups of population via statistical analysis (Dwork, 2011b). This protection can also encourage people to participate and give more honest answers, as the noise grants an objective level of maximum leakage, as well as a strong deniability against any conclusion that an adversary may extract from isolated records.

2.3.1 Formal Definition

Differential privacy aims to be a general approach to data privacy with high levels of protection, basing its definition on strong mathematical foundations. The original formal definition (Dwork et al., 2006) has appeared with different naming conventions and formats along the literature, being the one depicted

below extracted from the one used by Fanti, Pihur and Erlingsson (2016) because of its clarity. Let D and D' be two datasets differing in only one entry, $A()$ a generator of randomised responses and t a transcript of the database[1],

$$\Pr(A(D) = t) \leq e^{\epsilon} \Pr(A(D') = t)$$

The above formal definition imposes a restriction to randomisation mechanisms whereby an adversary would perceive a variation of a maximum of ϵ in the distribution of the database when only a transcript of the database changes (Dwork et al., 2006). In other words: In the case of a targeted attack, the probability of an adversary learning something from an user would only vary on a maximum of $\pm\epsilon$ with the presence or absence of that person in the database. This concept is known as ϵ-indistinguishability. The smaller the value of ϵ, the higher the privacy guarantee, as the two different datasets will be computationally closer. This requirement translates into noisier responses to queries about the data (Dwork & Roth, 2013, p. 5). The optimal value of ϵ seems to be a subjective matter, and no standard or average value seems to appear in the literature, where values ranging from 0.01 to 10 can be seen (Hsu et al., 2014).

A more relaxed definition was published on later works (e.g., Dwork & Smith, 2009; Dwork, Rothblum & Vadhan, 2010; Dwork & Roth, 2013, p. 18):

$$\Pr(A(D) = t) \leq e^{\epsilon} \Pr(A(D') = t) + \delta$$

This relaxation was considered due to the drastic utility reduction of the data after being altered via the original, more rigid definition (Kifer & Machanavajjhala, 2011; Huber, Müller-Quade & Nilges, 2013), where the difference between D and D' was bounded by ϵ in all cases. Thanks to this new definition, the possibility of a disclosure below ϵ levels transfers from impossible to improbable: Now, every query will be protected by the ϵ boundary in a minimum of $1-\delta$ cases out of 1 (Dwork & Roth, 2013, p. 18), which significantly increases the utility of population data, while still offering high levels of protection to individuals' privacy.

l_1−sensitivity

In order to design methods that generate noise to achieve differential privacy, it is necessary to define the sensitivity to noise of the function first. In a function

[1] An entry in the dataset (or database) is often referred in the literature as transcript(Dwork et al., 2006; Dwork, 2006), so from now on these terms will be interchangeable.

that outputs real numbers (e.g., count of number of rows that satisfy a condition, or fraction of these amounts), its sensitivity is defined as the maximum change that can be perceived between two adjacent databases (Dwork et al., 2006). The formal definition of this l_1-sensitivity is as follows:

$$\Delta f = \max_{D,D'} ||f(D) - f(D')||_1$$

The definition above describes Δf as the maximum amount of change that an individual's information can induce on the output of the query. For instance, a query that counts the number of individuals that own a car would have a sensitivity $\Delta f = 1$, while a query that groups individuals according to their favourite ice cream flavour would have a sensitivity $\Delta f = 2$, as the altered individual would change the amounts on the group that is leaving and on the group that is entering (Dwork et al., 2006). This defines the level of noise required to obscure the participation of any individual (Dwork & Roth, 2013, p. 31).

Once the sensitivity of the function is measured, it is possible to implement mechanisms that ensure $(\epsilon, \delta)-$indistinguishability or (ϵ)-indistinguishability. To demonstrate how these mechanisms can be built, the definition of the first proposed mechanism, which will be also mentioned in Section 2.3.2, is shown below.

Laplace Noise Generation Mechanism

Various mechanisms for noise generation are exemplified along the literature. Some of them, like the Laplacian , the Gaussian (Dwork et al., 2006) or the exponential (McSherry & Talwar, 2007) methods, follow a known mathematical distribution. Other methods exist for composing more elaborated distributions while still ensuring differential privacy (McSherry & Talwar, 2007). Each one of the methods possess qualities that make them useful for different cases.

In early stages, noise addition in the shape of a Laplace distribution was proposed (Dwork et al., 2006). This method was useful for low sensitivity functions, like counting values (e.g., most common first name (Dwork & Roth, 2013, p. 37)), but demonstrated to be too noisy for more complex queries, like correlating different variables (McSherry, 2016). This mechanism was formally defined as follows:

$$f(D) + (Lap(\Delta f/\epsilon))^k$$

Where k is the number of dimensions of the domain in the output of the function (as in $f : \mathbb{N}^j \rightarrow \mathbb{R}^k$) (Dwork, 2008).

A representation of the above formally described is made on the practical example made by Zumel (2015): First, Figure 2.1 shows the original outputs (green and orange) of a deterministic query (a statistical mean) according to two adjacent databases. According to Zumel, "set2" is represented upside down for clarity.

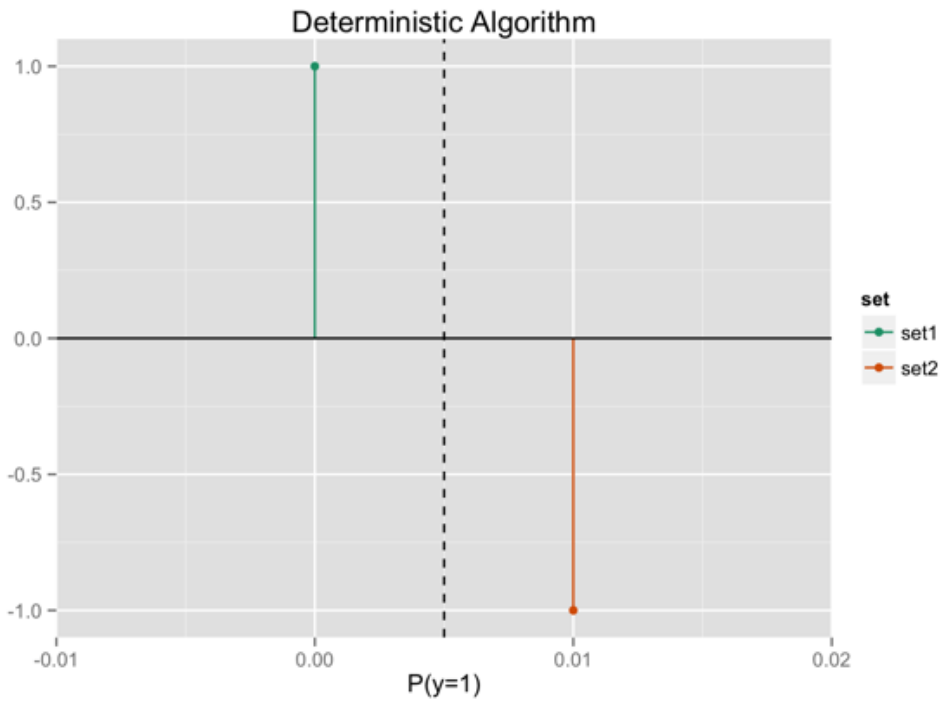

Figure 2.1: Original outputs without noise (Zumel, 2015)

The probability of "set1" returning a higher value than "set2" is 0 with the original data, but after applying noise following a Laplace distribution (Figure 2.2), an overlap between the two can be appreciated. The probability of "set1" returning a bigger result than "set2" is represented by the shaded green area, while the shaded orange area would be the opposed case. In other words, the difference between the orange and the green area represents ϵ[2].

[2]Concretely, the exact value of ϵ is found by performing $\epsilon = |\ln \frac{green}{orange}|$.

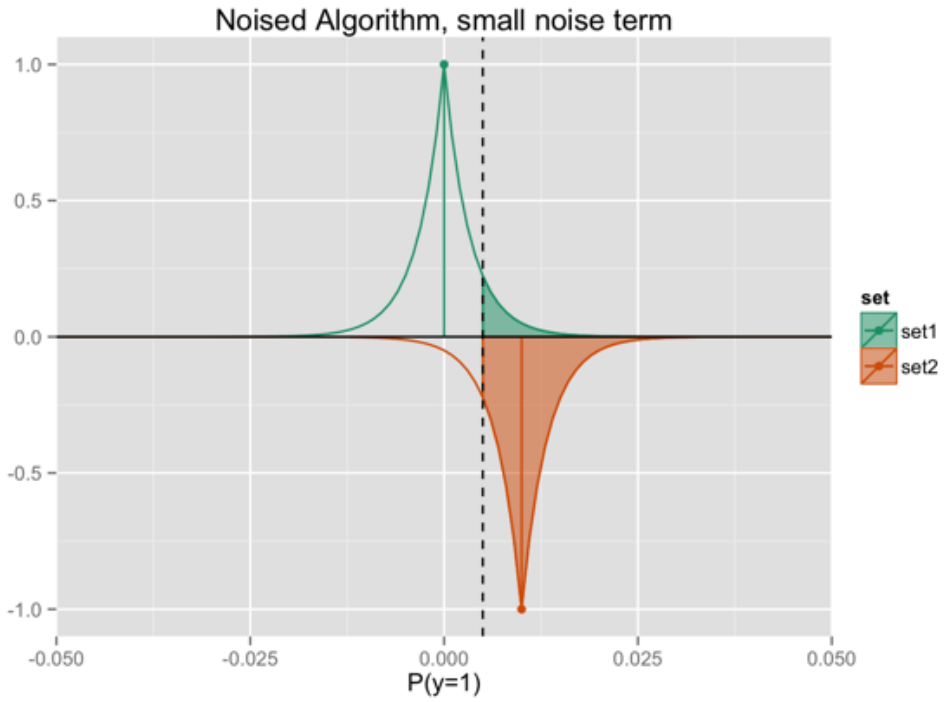

Figure 2.2: Low laplacian noise (Zumel, 2015)

The difference of probabilities is still too high for ensuring a sufficient level of differential privacy. By adding more noise, the Laplace distributions widen and overlap, reducing the value of ϵ, until the green and orange areas are almost the same, as Figure 2.3 shows.

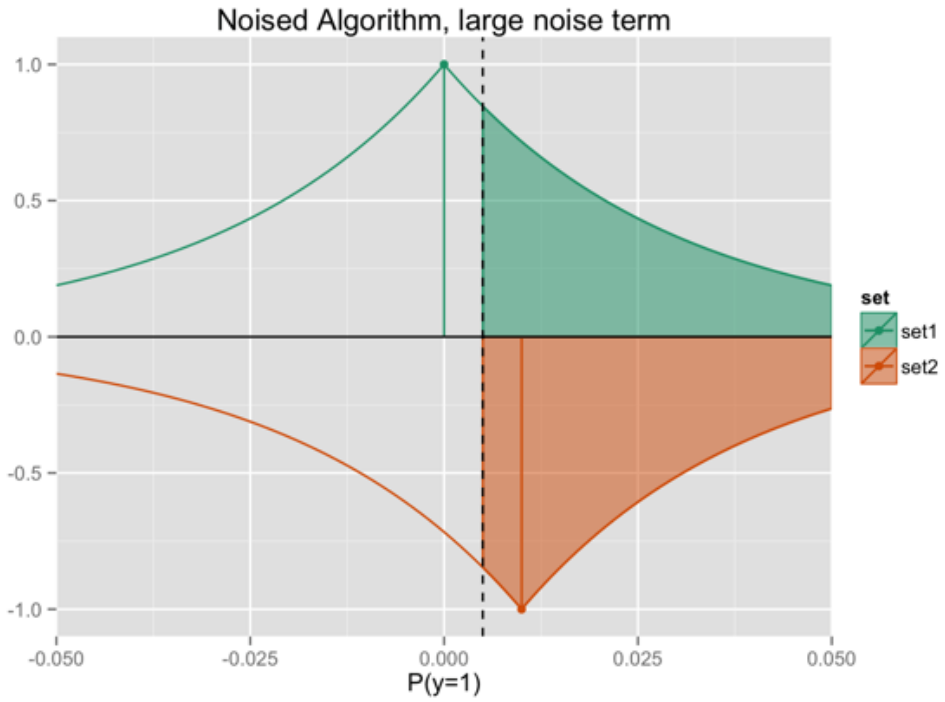

Figure 2.3: High laplacian noise (Zumel, 2015)

This example illustrates how the value of ϵ can be used to tune the level of noise of the output of the function, controlling its privacy guarantees and its utility.

2.3.2 Academia Support and Critique

The base provided by the theoretical work on differential privacy has been expanded over the years, enabling researchers to implement privacy-by-design applications in fields as diverse as genetics (Jiang et al., 2014), telecommunications (Hu et al., 2015), social networks (Giakkoupis et al., 2015), statistics (Wang et al., 2016; Machanavajjhala, Kifer, Abowd, Gehrke & Vilhuber, 2008), data science (Mohammed, Chen, Fung & Yu, 2011) or the Internet of Things (IoT) (Nguyên et al., 2016).

Although it is considered the gold standard of current privacy preservation research by authors like Hsu et al. (2014) or Huber et al. (2013), the parents of this method insist in saying that it is no panacea (Dwork, 2011a), and that each case must be studied individually for good tuning of the parameters. As an example of its recognition between scholars, one of the papers where the method was

initially presented (Dwork et al., 2006) was given the "Test-of-Time" award of the *Theory on Cryptography Conference* (Goldwasser, Ishai & Nielsen, 2016).

Only two pieces of academic work that posed arguments straightly against differential privacy were found. The first paper, published by Sarathy and Muralidhar (2009), assured to demonstrate how the previously described Laplace-based noise addition did not satisfy the requirements of differential privacy. As McSherry (2016) exhibits in correspondence with the authors, almost all their claims on that paper were based on wrong assumptions extracted from an erroneous copy of one of the core formulae that sustain differential privacy, concretely the l_1-sensitivity formula previously exposed in this work.

In 2014, Bambauer et al. published a considerably extensive critique against differential privacy, in which they presented various concrete examples where, according to them, differential privacy offers results that are whether too disclosive or exaggeratedly noisy. Again, McSherry explains in the same entry of his personal blog how the claims of Bambauer et al. are wrong from their conception or short-sighted, in the sense of they using wrong approaches and techniques without further justification.

An example of the first one would be the affirmation of that, in applying differential privacy, queries over small populations (or subsamples of big populations) will offer highly noisy results and, consequently, rendered useless (Bambauer et al., 2014, p. 19). Here, the critique from the authors is unjustified, as this absolutely follows the definition of differential privacy: As a small population is much more disclosive about its individuals than a large population, an adversary should learn less from the small one. Therefore, it can be extracted from this misunderstanding that differential privacy requires a large population to offer reliable results. The concrete size of this population and how this number relates with the level of privacy guarantee imposed remains uncertain.

McSherry also explains how arbitrary methods (e.g., statistical indicators, noise generation mechanisms) are used all along the text with no apparent reasoning behind their election, while appropriate tools already existed in literature that they themselves referenced. This is the case of employing an aleatory mechanism for noise generation (Laplace) described in one of the earliest papers, ignoring later research, or the usage of the average instead of the median for datasets with long-tailed distributions, which logically generated a distorted view of the population (Bambauer et al., 2014, p. 24).

2.4 RAPPOR

The Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR) is a mechanism based on differential privacy designed to collect statistics from end users while ensuring high levels of privacy (Erlingsson et al., 2014). As Erlingsson et al. state, RAPPOR arises from the necessity of service providers (e.g., Google, Microsoft, Apple, etc.) to improve their products according to the necessities of the population without affecting the privacy of the surveyed users, as it was discussed in Section 1.1.

It has been adopted into projects like the open source web browser Chromium and its Google variant, Chrome (Erlingsson, 2016; Chromium Developers, 2015) and the code is publicly available for anyone that desires to analyse it or use it on their own projects[3].

2.4.1 Architecture

RAPPOR is conceived as a client-server architecture where the response of the clients will be automatically randomised before being sent to the server to be aggregated with the rest of clients' responses (Fanti et al., 2016). This implies that, contrary to the main differential privacy authors clear inclination for interactive approaches (Dwork et al., 2006), RAPPOR is a non-interactive process, as the original data would not be recoverable. Justifications for this could be the easier implementation of this approach or the priority for data security obtained from never handling real data.

2.4.2 Algorithm

Encoding

The information of the reports is produced as strings, which are encoded into Bloom filters (Bloom, 1970) that allow to collect arbitrary strings for their later comparison with candidate strings (Erlingsson et al., 2014). This also enables to reanalyse the collected reports against strings newly learnt by the data controller without loss of privacy. It is over these Bloom filters that the noise will be added.

[3] http://github.com/google/rappor

As an example, Figure 2.4 shows how the string “Dog” is encoded into a Bloom filter of 8 bits with 3 different hashes (each hash will set only one bit of the array, and the amount of hashes employed is determined by the variable h).

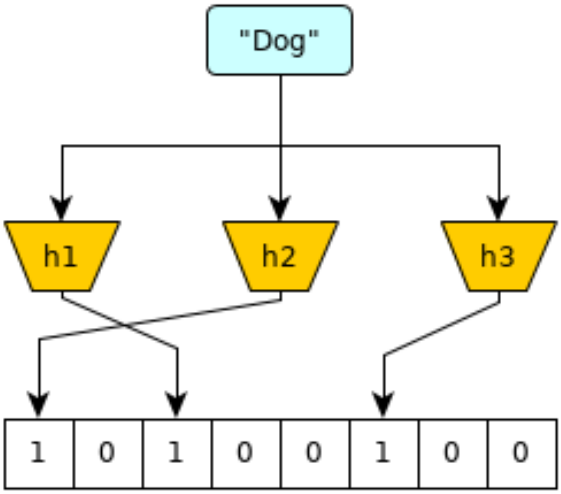

Figure 2.4: Example of a Bloom encoding with 3 different hashes

2.4.3 Noise Addition

Noise is added to the Bloom filter following the surveying technique of randomised response (Erlingsson et al., 2014). This theory first proposed by Warner (1965) was designed to encourage participants in a survey to answer more honestly to sensitive questions: A coin was flipped to decide whether the answer to the question would be automatically “yes” or the honest response, providing a strong deniability to such answer. Probability says then that half of the population would have answered “yes” regardless of the truth. Assuming that the other half of the answers were truthful and that the proportions in the population are represented accurately enough on the sample, the real proportion of “noes” would be the double of the proportion obtained from the survey.

After the Bloom filter is generated, two different randomisation phases are used to generate the final noisy response.

Permanent Randomised Response

Before starting this phase, the original Bloom filter is searched into the registry of already generated filters. The literature employs B for the Bloom filter, and B' for the PRR (Fanti et al., 2016; Erlingsson et al., 2014), so this denomination will be used from now on for the sake of simplicity. If B' was already generated for this B, it will be used again and this phase will be skipped (Fanti et al., 2016). This is done to ensure that an attacker will never have a variety of B' for the same B and user, which would enable them to learn the real value (Erlingsson et al., 2014).

The PRR is generated according to a user-tunable variable called f (Erlingsson et al., 2014). B' will be generated as follows:

- B'_i will be 1 with a probability of $\frac{1}{2}f$
- B'_i will be 0 with a probability of $\frac{1}{2}f$
- B'_i will be B_i with a probability of $1-f$

Instantaneous Randomised Response

After generating (or retrieving) B', an IRR (denominated as S) will be produced using the variables p and q. This way, there will be only one B' for the same original data, but a different S will be generated every time that the same piece of data is sent (Erlingsson et al., 2014). The mechanism for generating S is shown below:

- S_i will be 1 with a probability of q if $B'_i = 1$
- S_i will be 1 with a probability of p if $B'_i = 0$

Figure 2.5 shows a simplified version of the final report generation process starting from a string.

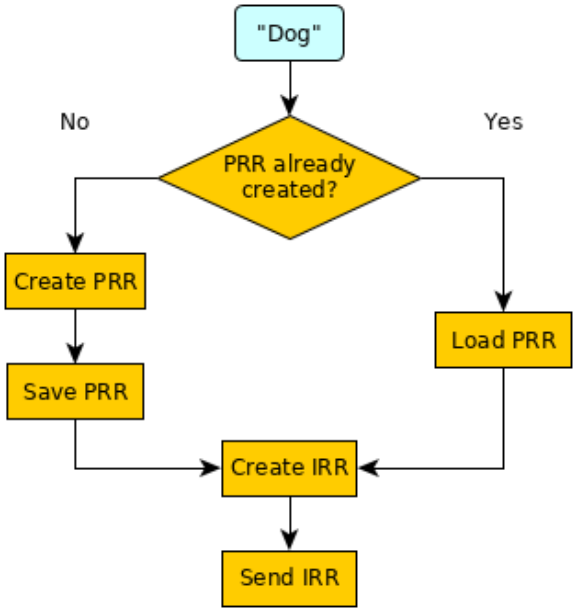

Figure 2.5: Basic flowchart of the RAPPOR algorithm

2.4.4 Modifications

The process of RAPPOR can be altered so that it suits different situations where more or less aspects of security are needed. As an example, Erlingsson et al. suggest three simplifications for different collection scenarios.

One-time RAPPOR

In a case where every individual will be surveilled only once, the IRR becomes unnecessary, as no targeted attacks against only one user would be possible. Setting $q = 1$ and $p = 0$ would result in skipping the IRR phase.

Basic RAPPOR

When the range of expected responses is considerably small and known beforehand (e.g., yes or no, male or female, etc.), every response can be mapped to a single bit of the Bloom filter, making it easier to recover information from populations due to the lack of collisions. This would be made using just one hash on the filter ($h = 1$).

Basic One-time RAPPOR

In the simplest of the scenarios, where only one response will be collected for each user and the amount of possible responses is small enough to map each response to a single bit, a combination of the two previously mentioned simplifications can be used.

2.4.5 Differential Privacy on RAPPOR

Erlingsson et al. (2014) formally demonstrate the compliance of RAPPOR with differential privacy. The PRR ensures a level of $\epsilon_\infty-$indistinguishability of:

$$\epsilon_\infty = 2h\ln\left(\frac{1-\frac{1}{2}f}{\frac{1}{2}f}\right)$$

Adding the IRR phase implies that the probability of knowing if there was a bit set on the original filter by observing the final report is altered by both the IRR and the PRR phases. The probability of observing a 1 on the final report when there was a 1 set on the original filter is described as:

$$q* = P(S_i = 1|B_{i=1}) = \frac{1}{2}f(p+q) + (1-f)q$$

While the probability of observing a 0 on the final report given that the original report the bit was set to one is:

$$p* = P(S_i = 1|B_{i=0}) = \frac{1}{2}f(p+q) + (1-f)p$$

These two probabilities are used to calculate the final differential privacy level of the RAPPOR configuration[4]:

$$\epsilon_1 = h\ln\left(\frac{q*(1-p)}{p*(1-q)}\right)$$

In order to verify the calculations made by Erlingsson et al. and as an utility to further practical works, a python script that calculates ϵ_∞ and ϵ_1 can be seen on Appendix A.

[4]On Erlingsson et al. the formula appears with "log" instead of "ln", but after testing the values they obtain for ϵ_1, they match the calculations made with Napierian Logarithms.

2.4.6 Limitations

The original implementation of RAPPOR has its design limitations, some of them are due to the lack of abundance of collaborations in the project, and others are rooted in its own design.

Correlative Reports

Records that can be correlated over time (e.g., asking users every day to provide their age in days) can be predicted and will reduce the levels of differential privacy (Erlingsson et al., 2014). Given this scenario, additional security measures should be taken in consideration, as increasing the level of noise over time.

Variables cannot be Correlated

As the values more present in the population were learnt a posteriori, correlating the occurrence of different values (e.g., people who smoke have a higher chance of having cancer) was not possible when the work of Erlingsson et al. (2014) was published. Fanti et al. (2016) proposed the utilisation of the Expectation-Maximisation algorithm (Dempster, Laird & Rubin, 1977) to estimate the correlation between distributions of two different variables. A demonstration of the code built can be seen in RAPPOR's Github repository.

Candidate Strings must be Known Beforehand

The learning mechanism initially proposed by Erlingsson et al. relies on a list of candidate strings to match against, which limits the usability of RAPPOR (e.g., if the data collector asks for users' homepages, the websites not present in the dictionary will not be detected, even if some of them appear in big proportion). To solve this issue, Fanti et al. propose a new technique to generate a list of candidate strings based on the collection of small substrings of n length (called $n-$grams) that can later reconstruct the candidate strings. Again, a sample of this process was included to the project's Github site.

2.5 Conclusions

Evincing the necessity of data collection by companies and the expectation of this tendency to keep growing in the future, different classifications have been exposed in an attempt to catalogue all the different alternatives that could enable this activity while preserving the privacy of the users.

Along the literature review, diverse options for privacy preservation have been presented and detailed. Advantages and disadvantages have been objectively discussed for every method, providing of a general understanding of the scenario in privacy preserving mechanisms.

From the three different categories exposed, it seems that only the randomisation method remains as a solid alternative: Generalisation has followed a thorough cycle of refutation and amendment that does not seem to come to an end, as even the newest proposals of this technique show disclosure or usability problems (Domingo-Ferrer & Torra, 2008). Query auditing, on the other hand, seems to have lost relevance in the academic world over the years, with diverse scholars considering it unfeasible or disclosive by design (Dwork, 2011b; Kenthapadi et al., 2005).

From the randomisation category, differential privacy was chosen in particular as it was based on solid mathematical foundations (Dwork, 2006; McSherry & Talwar, 2007), and relevant firms like Google, Microsoft or Apple have started to get interested in this concept in the late years. The section showed how this method provides an objective measurement of noise addition techniques and enables the fine tuning of such noise. This, added to the fact that no strong arguments against the definition have resulted to be consistent enough to disprove it, seem to be the main drivers that keep encouraging researchers to widen the area of investigation on this regard.

The only downside of this proposal seems to be the need of a broad understanding of its inner functioning in order to properly configure its parameters, ensuring a sufficient level of privacy guarantee without preventing data controllers from retrieving usable information. The scarce work on documenting and standardising procedures that ensure these conditions (Hsu et al., 2014) could be the main reason behind the slow transition of this theory from academy to industry.

A mechanism known as RAPPOR (Erlingsson et al., 2014; Fanti et al., 2016), developed by Google and included in the Chrome web browser for population data collection, is studied as an implementation of differential privacy. This

open sourced technology (the only one that seems to be publicly available) is accompanied by two academic papers that thoroughly explain its inner functioning and expose empirical tests to demonstrate its efficacy. Some technical limitations were also highlighted, as the difficulty to correlate detected values or the weakening of the privacy guarantee as report values correlate over time.

The tests performed by Erlingsson et al. (2014) show how the noisy responses can be used to retrieve the original data with high accuracy, given enough data collected. Nevertheless, no further analysis of this evolution of the fidelity of the data is offered, apart from graphical comparisons of the original distribution against the retrieved ones. A more formal description of the similarities and differences of those distributions would help to further understand how RAPPOR behaves with different input values.

Chapter 3

Methodology

3.1 Introduction

This chapter depicts the procedure that will be followed in order to perform a series of experiments over the RAPPOR platform that illustrate the effects of the variation of different initial parameters over the final results.

Section 3.2 reflects over the findings made on Chapter 2, arising facts that contradict each other or remain unclear after the analysis of the literature.

Section 3.3 compiles the aforementioned thoughts into research goals that will determine the design of the experimental phase.

Section 3.4 contains the details of the experiments that will be performed. It shows the reasoning behind the election of the data over which the experimentation will be made, and the transformations required to completely adjust the format of the dataset to the requirements of the platform. It also justifies the election of the different values for ϵ and the sizes of the population that will be chosen for the practical part of this work. Finally, a discussion about the possibilities availables for analysing and evaluating the data that will be obtained from the experiments is made.

3.2 Research Questions

Taking the work of Erlingsson et al. (2014) as a reference, it can be extracted that, for $\epsilon = \ln(3)$, a dataset of 10.000 answers will be insufficient to extract useful information. As it could be expected from the literature review, a higher value of ϵ would guarantee a more similar resemblance between the recovered information and the original data. Furthermore, the election of $\epsilon = \ln(3)$ is not justified at any point, remaining unclear if this level of protection is simply appropriate for testing, acceptably secure, or simply insufficient. A question that naturally arises from this fact is how different values of ϵ affect the accuracy of the data.

As Hsu et al. (2014) state, a procedure for selecting an optimal (or even sufficient, or non excessive) value for ϵ is something that is still missing in the literature. This absence is, at least partially, due to the difficulty of estimating an optimal value for a single variable that must consider factors like the size and diversity of the dataset, the sensitivity of the information contained inside this, or the level of accuracy required for the reconstruction of the noisy response. In synthesis, the main disadvantage of the simplification of the scenario is the complexity of its optimisation.

Considering this situation, it appears to be evident that elaborating a general rule for predicting how the variation of ϵ will affect the privacy and fidelity levels of a specific dataset is something far from simple. An empirical demonstration of the progressive quality degradation of the recovered data as ϵ decreases could clarify this matter and help other researchers willing to implement differential privacy on other projects to choose appropriate configuration values.

Another statement extracted from Erlingsson et al. (2014) is the loss of information caused by an oversized dataset, where smaller real values can be obfuscated by the noise due to their proportionally reduced presence, creating a situation where, literally, "more is less". In spite of being a logical conclusion, the affirmation is not supported by empirical data, which leads to wonder how and where that inflexion point could be found.

3.3 Experimental Aims

To summarise the reflections above discussed, the following statements have been formulated in order to determine specific objectives to seek during the development of the experimental phase and evaluate it afterwards.

3.3.1 The Value of ϵ

Decreasing the value of ϵ will offer noisier results, these not even being useful if the amount of data collected is not enough. Understanding the effects of the variation of this parameter over the retrieved data is key to design a procedure that offers reliable information to the data controller while protecting the privacy of the users.

3.3.2 The Size of the Dataset

For the same value of ϵ, datasets with different sizes will be affected in varied manners. Comparing results over different populations can help to estimate the minimum number of reports needed by different levels of privacy guarantee, as well as to anticipate suitable configurations for concrete practical cases. If the results obtained are comparable to the ones offered by Erlingsson et al., this will also verify the procedures followed by them.

3.4 Experiment Design

The experiment proposed in this work attempts to design different scenarios that will allow to study the effects produced by the variation of one of the two input parameters. The main idea is to observe differences between original data and data recovered from the RAPPOR reports, and also between data recovered from different scenarios (i.e., scenarios with the same number of reports but different values of ϵ and vice versa).

3.4.1 Dataset

Originally, it was considered to include a first stage of data collection and report generation for this project, but the fact that RAPPOR requires a dataset with a considerable size and a wide variety of participants made this phase impracticable due to the amount of time available for the project. Instead, it was decided to use an already available dataset.

Some candidate datasets were considered, prioritising factors like similarities with the data used by Erlingsson et al. or relevance of the data in terms of security and privacy. Also, special interest was put on finding data related to smartphones, as these devices contain significant amounts potentially sensitive information about their users, and a considerable amount of data is collected from them on a regular basis (Lord, 2012). The work of Frank, Biedert, Ma, Martinovic and Song (2013) was considered for fitting into several aspects of the previous description, but it was discarded for being based on numerical data, which the current implementation of RAPPOR cannot handle properly (Fanti et al., 2016).

The chosen candidate was a smartphone dataset destined to security research (Mirsky, Shabtai, Rokach, Shapira & Elovici, 2016). This dataset collected in-

formation about the devices of the participants at the same time that a fake piece of malware was leaving traces, being the general idea to study methods for detecting unusual behaviours that could indicate the presence of a piece of malware.

The election of this dataset was caused by the presence of process names as one of the pieces of data collected, establishing similarities between this experiment and the one performed by Erlingsson et al. (2014). Process names is a suitable parameter for RAPPOR, as it translates into a set of strings, some of which should have a very strong presence (e.g., system processes, common applications) while others should scarcely appear.

Access to the whole dataset was requested, and although there was positive feedback to the application, no further notifications were received from the research team. Nevertheless, a free sample of such dataset was offered in the project's website[1]. Even though this sample only contained information about a single user, it still qualified to be used for the purposes of this work, as it will be explained further on. Figure 3.1 shows that the sample contained more than 14 million records, all belonging to the user "97bb95f55a".

```
[case@archimedes data] >> wc -l Applications.csv
14801900 Applications.csv 1
[case@archimedes data] >> awk 'BEGIN{FS=","}{print $1;}' Applications.csv | uniq
UserId
97bb95f55a 2
```

Figure 3.1: Commands showing the amount of reports in the sample (1) and the only existent user (2)

Redoing the experiments with the entire dataset was considered after access was granted, but due to time constrictions and lack of computing power and storage capacity (the full dataset occupied more than 600 GB), the idea was dismissed.

Alteration of the Dataset

In spite of the lack of variety of users, with more than enough pieces of data, this sample has still the potential of being used for the experiment, if some adaptations are made before.

[1] http://bigdata.ise.bgu.ac.il/sherlock/index.html

The field *UUID*, which represents the UNIX millisecond timestamp of the report's collection (Mirsky et al., 2016), could qualify as a field by which the records could be grouped by, substituting the user. As Figure 3.2 shows, more than 300 000 users could be obtained using the *UUID* field (a considerable quantity, compared to the 10 000 users from the second experiment made by Erlingsson et al.), representing each user as a one-time collection of the status of the phone.

```
[case@archimedes data] >> awk 'BEGIN{FS=","}{print $2","$5;}' Applications.csv | head -10
UUID,PackageName
1461792776014,com.google.android.gm
1461792776014,flipboard.app
1461792776014,com.google.android.apps.plus    1
1461792776014,ch.smalltech.battery.free
1461792776014,com.google.android.inputmethod.latin
1461792776014,com.android.systemui
1461792776014,com.google.android.talk
1461792776014,com.android.systemui
1461792776014,com.android.server.telecom
[case@archimedes data] >> awk 'BEGIN{FS=","}{print $2;}' Applications.csv | sort | uniq | wc -l
307051    2
```

Figure 3.2: Commands showing the appearance of the columns UUID and PackageName (1) and the amount of different UUIDs present on the sample (2)

Offering around 46 reports per user, compared to the 18 of Erlingsson et al., this sample should be more than enough to obtain accurate results from RAPPOR, given the same parameters. It could be argued that this adaptation would change the meaning of the data, but the experiment seeks to observe differences in the distributions of recovered data, not to learn from it.

Aside from the two attributes shown in Figure 3.2, the rest of the data was obviated for not being necessary for the experiment, making it closer to the one it is aimed to resemble to. It is true that a more human-readable output could be obtained from column "ApplicationName" than from "PackageName", but as Figure 3.3 shows, some applications contained strange characters that could complicate the process, so the "PackageName" attribute was chosen instead.

```
[case@archimedes data] >> awk 'BEGIN{FS=","}{print $3","$5;}' Applications.csv | head -200 |grep "pango\|Appl"
ApplicationName,PackageName
ﻮﺠﺠﺍ+,com.unicell.pangoandroid
ﻮﺠﺠﺍ+,com.unicell.pangoandroid
ﻮﺠﺠﺍ+,com.unicell.pangoandroid
ﻮﺠﺠﺍ+,com.unicell.pangoandroid
```

Figure 3.3: Non alphanumeric characters shown on the "ApplicationName" column

3.4.2 Election of Population Sizes

In their first experiment, Erlingsson et al. used three different populations of 10 000, 100 000, and 1 000 000 responses to demonstrate the effect of the amount of the reports collected on the quality of the retrieved information. By using similar sizes, this experiment will allow to verify their empirical work.

Looking at the dataset that was chosen, even just extracting one report from each user would produce a sample of 307,000 records (Figure 3.2). In order to produce smaller samples, a selection of users will be chosen to obtain a reduced amount of records.

3.4.3 Election of ϵ Values

As it was mentioned in the literature review (Chapter 2), values on the bibliography range from 0.01 to 10 (Hsu et al., 2014). The value chosen by Erlingsson et al. is $\epsilon = \ln(3)$, which could be established as the intermediate value. As the experiment aims to be illustrative about the changes suffered between these variations, the high value will be extracted directly from the upper end (10) of the previously mentioned range in the literature, while the low value will be 0.1. Tests were run using 0.01, but no results at all were retrieved (0 detected occurrences of each candidate string).

This fact is illustrative by itself on how low that value of ϵ may be for the population sizes chosen, but the same conclusions can be reached by using $\epsilon = 0.1$ (as it will be seen in the sections to come), which also offers more information to be discussed. Choosing 0.1 over 0.01 also offers an interesting point of view for the experiment, as the proportion between adjacent values would be always 1:10.

3.4.4 Evaluation of Results

As it was previously discussed, the original work of Erlingsson et al. does not offer detailed information about the results of their experiments, apart from the graphical comparisons between the original and the recovered distributions.

The type of data that the experiment will offer is a discrete count of appearances of each string. Ideally, a reliable statistical method should be used to evaluate the quality of the data obtained from the experiment. Unfortunately, the time frame in which the project was developed did not allow proper researching into

such methods and the theory behind them. Instead, a simpler method of evaluation that aims to be as objective and rigorous as possible is proposed:

Apart from the raw number of retrieved strings, each of the scenarios will be evaluated according to the proportion of recovered strings that were detected with, at least, 80%, 90% and 95% of accuracy. The first threshold represents the recovery of a string count with a sufficient level of precision, while the other two represent the amount of results that were retrieved with high precision.

The proportions are represented in comparison to the total of the strings detected, and not to the total of the strings present in the original data. As in a real use case the actual information of the population would not be available, it was considered more illustrative to represent the information in this manner, as it depicts how much of the retrieved information is indeed reliable.

By comparing these figures, the methodology above described aims to offer a more objective comparison of the different distributions, providing additional parameters that can be measured and discussed.

3.5 Conclusions

Starting with a thorough reflection on what was learnt from the literature review, this chapter has defined the parameters under which the experimentation phase will be effected. This critical analysis over the unclear areas left after the preliminary research led to two different aims to be sought in the following chapters: Studying the effect of varying the value of ϵ over the same population, and evaluating how the same parameters for RAPPOR may affect populations of diverse sizes.

To accomplish these objectives, a suitable dataset that enabled to compare the results with the ones obtained by Erlingsson et al. was chosen. The selected candidate featured process names as one of the parameters collected from the study, in the same manner that the original RAPPOR paper did.

Even though the dataset had to be altered to meet the requirements of the experimental phase, the fact that the information about a single user was transformed into multiple users by grouping the reports into simulated users does not affect the veracity of the results that will be obtained: The aim of the experiments made in this work strives on observing differences between the original data and the recovered distribution. The original meaning and repercussions of data itself should not affect the execution nor the analysis of the tests.

The values for the different values for ϵ were chosen according to the literature. The lowest level was increased due to null results for $\epsilon = 0.01$, which, although revealing, did not provide the same amount of discussion material as $\epsilon = 0.1$, which can be used to reach the same conclusion (i.e., these levels of indistinguishability are too demanding for the chosen sizes of population) while adding some additional perspective.

The evaluation method chosen aims to be simple and easy to implement at the same time that remains objective and reasonable. Pursuing more thorough statistical methods was dismissed because of time restrictions and a lack of elevated knowledge in the area.

Chapter 4

Implementation

4.1 Introduction

The following chapter shows the execution of the empirical steps that will lead to the final conclusions of the work, together with the detailed explanation of the development of the tools required to perform those experiments.

Section 4.2 analyses the code offered by Google to enumerate the different necessities that the experimental pipeline must cover to satisfy the requirements of the modules. First, a critical examination of the status of the code repository that the researchers offer is done, highlighting differences between the description of RAPPOR made in literature and the implementation made on the code. Next, the files that the analysis application requires are examined and described.

Section 4.3 contains the design of the final pipeline of modules that will produce the files to be analysed by the application later on, backtracing the operations needed to obtain them and determining the initial and intermediate files to produce.

Section 4.4 details the operations taken to adapt the data of the dataset to the format required by the modules of the pipeline. It also explains the subsampling operations effectuated over the original data to obtain the samples over which the experiments will be performed.

Finally, Section 4.5 describes the execution of the experiments, enumerating the final parameters chosen for the creation of the different scenarios and showing the process for generating the experimental data required later evaluation.

4.2 Code Analysis

4.2.1 GitHub Repository

An early analysis of RAPPOR's repository[1] showed that, apart from the core software for the server side of RAPPOR organised in a set of Python scripts that called R and C++ programs, other sets of utilities like client libraries, scripts for server automation or web applications for easy simulations and data analysis were also present. A further inspection of the documentation (under the path `doc/data-flow.md`) revealed a highly detailed description of the data needed and the modules to run on them to obtain reports from original data, and a distribution from the reports.

Unfortunately, a check of the pipeline described in the documentation (all scripted inside `demo.sh`) revealed that some modules have been changed since the documentation was created. Due to this issue, a closer analysis of the code was required to establish a pipeline and produce the data required for the experiment.

4.2.2 Inconsistencies between Literature and Code

Lack of Memoisation on the PRR

After inspecting the client code provided by Google, it was discovered that the generation of the PRR does not work exactly as described in the work of Erlingsson et al. The memoisation phase is nonexistent, as the PRR is produced via a deterministic algorithm that will output always the same result for the same parameters. There is no randomisation in this process.

This change of design alters the functioning of the process, but is still valid for generating the reports, as it relies on the f value to control the noise added to this phase. The main disadvantage is that, given that an attacker captured the parameters "userSecret" and "encoderID" from a targeted user, a crafted dictionary could be constructed to decode the answers of this individual. On the other hand, avoiding the necessity of storing the memoised answers could be a safer approach, as this database of PRRs would also give a dictionary to the attacker, and in this case it would be a dictionary of real responses, instead of just candidates.

[1] https://github.com/google/rappor

Maximum Size of the Bloom Filters

Both the Python and the C++ clients offered in the repository possess a limit of 32 bits while configuring the size of the Bloom filter to generate the reports. In spite of this, according to the experiments shown on the original paper, the sizes of the Bloom filters used in them were considerably larger:

The first experiment uses Basic One Time RAPPOR, which requires an individual bit for every candidate string, and considering that 200 candidates were employed, and that the size of the filter has to be a number such as $x = 2^n$, the minimum size of this filter had to be 256 bits. The second, third and fourth experiments used Bloom filters of 128 bits.

This necessarily implies that the code used by Erlingsson et al. differed from the one offered in the GitHub repository, and makes it impossible to reproduce their experiment.

A Java client was also included in the repository, and this one allowed larger filters. Nevertheless, the method for generating the hashes was different in this client, rendering the hash map to useless. Modules for massive reports generation and for hash map generation were developed using this Java client and included in the pipeline, but the results obtained with them offered no consistency, as it can be seen on Figure 4.1.

In spite of this, according to the experiments made by Erlingsson et al. (2014), the size of the Bloom filter should not affect the retrievability of the data, and it does not alter the value of ϵ. A 32 bits Bloom filter with 2 hashes offers $32^2 = 1024$ different combinations, more than enough compared to the 154 candidate strings present in the experiment.

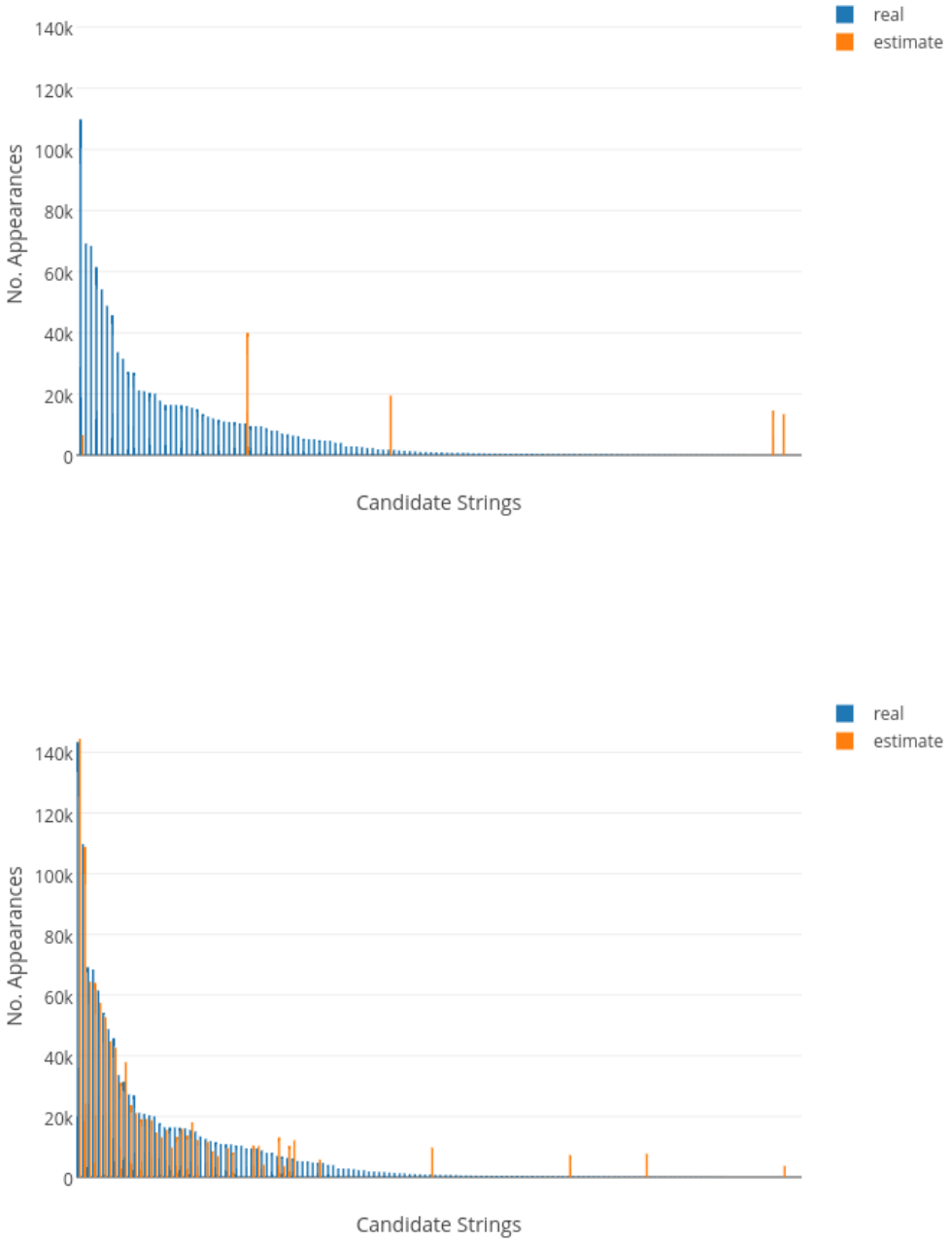

Figure 4.1: Data recovered with the Java implementation (up) versus data recovered with the already existent Python code (down)

The above graphs represent two tests made over the 1 200 000 reports dataset with the pipeline that employs the modules provided by Google (down) and a pipeline where the creation of reports and hash map modules were replaced by the Java modules that can be seen on Appendix F (up). The tests were made with 32 bits Bloom filters, 4 hashes, $p = 0.5$, $q = 0.75$ and $f = 0.5$.

The impossibility of replicating the levels of quality in the retrieved data prevents from using the Java modules and, therefore, from using Bloom filters greater than 32 bits. Nevertheless, the usage of enough number of hashes enables to create a sufficient number of combinations for the candidate strings, and will not affect the variations expected to be seen in the experiment, as it will be shown later on in the text.

4.2.3 Files Required

Executing the web application that enabled the analysis of the RAPPOR reports (`apps/rappor-analysis/run-app.sh`) helped to understand the data required to perform the analysis. As Figure 4.2 shows, three different files called "params", "counts" and "map" were required to perform this analysis.

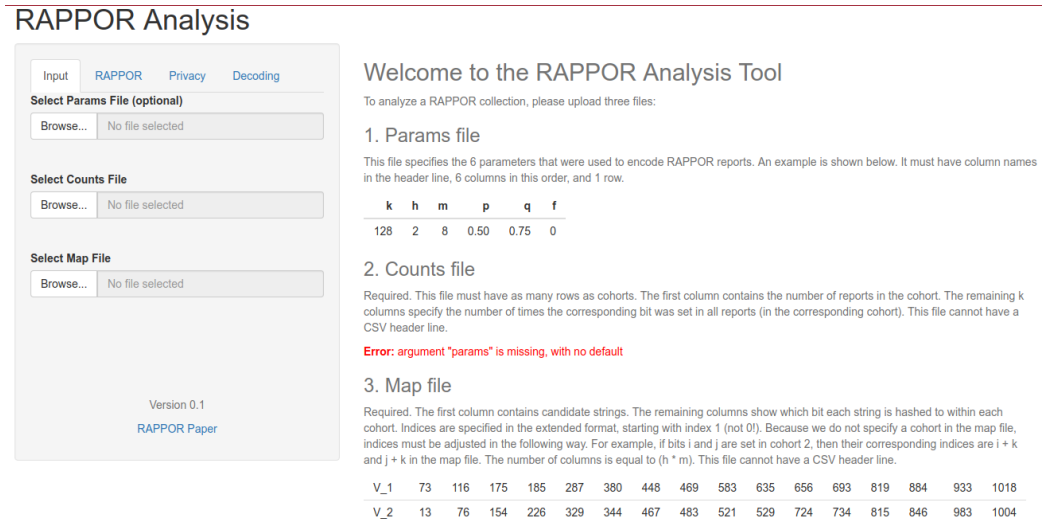

Figure 4.2: Interface offered by the web server for analysing the RAPPOR reports

Running the demo (`demo.sh`) produced the mentioned files, located under the paths `_tmp/python/demo3/case_params.csv`, `./_tmp/python/demo3/case_map.csv` and `./_tmp/python/demo3/1/case_counts.csv`. The structure of such files is as follows:

Params

A simple file that describes the parameters configured in the RAPPOR client while generating the reports. The first row is a header that determines the variables for the size in bits for the Bloom filter (k), the number of hashes used to encode the response (h), the amount of cohorts in which users were grouped by (m), and the noise generation parameters of the Permanent Randomised Response (p) and the Instantaneous Randomised Response (p,q). The second row contains the values of these variables. An example of such file can be seen on Figure 4.3.

	A	B	C	D	E	F	G
1	k	h	m	p	q	f	
2	32	1	64	0.25	0.75	0.5	
3							
4							
5							

Figure 4.3: File params.csv obtained from the demo

Counts

A headless file with a row for every cohort (64 in this example). The first column contains the amount of reports retrieved for that cohort, and the rest of columns show the sum of all the bits in that position (the sum of the bits activated in the first position of the filter will be on column number 2, and so on). This generates a table of m rows and $k+1$ columns. An example of this table can be seen on Figure 4.4.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	15620	5882	5939	5931	5892	6216	5793	5922	6199	5812	5798	5812	6183	5887	5979	6167	6095	5919
2	15630	6200	5825	5903	6132	5909	6193	5906	5894	6067	5940	5950	5997	6017	6130	5846	6083	5772
3	15630	5992	5885	5846	6089	5990	5960	5865	5845	6023	5971	5993	6089	5945	5855	5984	5937	6053
4	15630	5928	5962	5885	6081	5789	6235	6020	6028	6024	5950	5919	5747	5940	6021	5896	6123	6007
5	15630	6088	5797	6004	5943	6049	5989	5962	6252	5892	6065	5944	6087	5888	5999	5771	6214	5950
6	15630	6114	5850	6354	5738	5900	5974	6260	5983	6031	5951	6217	5854	5992	5956	6144	5874	6021
7	15630	6017	5865	5997	6057	6088	5952	5934	6070	6141	6341	6061	5854	5891	5796	5845	5940	5949
8	15630	5992	5984	6012	5980	6004	6007	5968	5916	5802	5992	5989	6055	6029	5767	6121	5923	5824
9	15630	5987	5915	6028	5765	5898	5995	6008	6002	6002	6166	5946	5985	5862	6288	5882	6044	5993
10	15630	5933	5847	5937	5841	6133	5871	6143	5915	6353	5846	5833	6052	5921	5871	6467	6211	6034
11	15630	5962	6001	5878	5993	5916	5908	5924	5901	6048	6026	6089	5852	5886	6103	5875	5837	5937
12	15630	5908	6171	6102	5841	5948	5690	6104	5964	5898	5880	5970	5799	5997	6137	5851	5956	5783
13	15630	5979	5871	5951	5950	5810	5988	5833	5865	5897	6103	5846	5974	5872	5989	6203	6251	5930
14	15630	6012	5835	6157	6071	5938	5828	5947	6194	6147	5902	5907	5918	6015	5909	5925	5930	6102
15	15630	5958	5804	6305	5793	6125	6317	6076	5980	6154	6056	5957	5967	5858	6038	6083	5952	5944
16	15630	5822	6057	6054	6020	6019	5976	5880	6166	6026	6123	6075	5950	5878	6013	6188	5970	5973
17	15630	5969	6003	6139	6061	5931	5906	5902	6242	5940	6096	5973	6032	5959	5931	6095	5981	6108

Figure 4.4: File counts.csv obtained from the demo

Map

Another headless file that contains a list of the candidate strings to find on the reports. Each row corresponds to a different candidate string. The string itself is shown in the first column, while the rest of the columns contain the position (starting from 1) where the bit is set in the original Bloom filter for that string in the determined cohort for a concrete hash, producing a total amount of $m*h+1$ rows. This configuration can be seen on the file shown on Figure 4.5.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	v1	25	36	83	100	144	171	217	242	273	304	329	354	405	446	451	495	542
2	v2	8	33	91	116	129	161	197	243	285	309	350	377	406	441	453	490	522
3	v3	15	51	89	102	155	163	202	237	270	303	350	366	400	434	454	500	534
4	v4	12	59	85	104	157	171	213	254	266	303	338	359	400	421	451	488	515
5	v5	30	51	82	112	158	191	218	242	272	295	341	376	413	419	455	512	536
6	v6	5	41	73	120	138	167	196	251	279	310	352	376	394	430	462	505	527
7	v7	27	53	96	102	136	180	202	242	286	320	334	381	412	426	449	482	543
8	v8	31	55	68	110	129	163	202	243	270	303	322	378	399	437	459	512	529
9	v10	8	57	70	128	154	168	212	225	272	319	326	360	385	435	480	484	520
10	v11	1	45	84	121	135	167	214	256	286	295	330	377	412	432	480	491	530
11	v12	23	48	87	103	142	180	212	234	265	316	340	380	414	439	454	509	540
12	v13	19	48	80	117	147	179	201	239	262	293	324	365	410	443	458	512	516
13	v14	16	51	81	105	157	190	223	252	275	310	346	378	411	429	454	485	536
14	v15	27	46	96	121	137	190	195	243	277	305	343	378	407	435	472	496	533
15	v16	20	46	65	106	159	175	193	236	264	297	337	355	390	431	476	491	534
16	v17	31	38	79	103	132	191	201	244	257	291	351	375	415	434	456	502	540

Figure 4.5: File map.csv obtained from the demo

4.3 Pipeline Design

In order to backtrace the generation of the files and design a sequence of actions that allows to obtain them from the original samples, the previously described demonstration script was analysed. The script `demo.sh` resulted to be just a wrapper that called `regtest.sh` with different sets of parameters. This last file contained a full pipeline completely automated, making calls to diverse modules that generated an artificial dataset and processed it until it generated the final results. The analysis of this file allowed the design of a custom pipeline to perform the intermediate steps that would output the required data for its analysis. A description of the full process is depicted on Figure 4.6.

The step from the dataset to the `true_values.csv` file was implemented via a Python script that can be seen on Appendix B, and the scripted pipeline that produces the necessary files for the analysis has been included on Appendix D. The intermediate files required for this task are described below.

The final output of this pipeline is a graph that compares the original data with the values recovered from the RAPPOR process. This graph is plotted from a file (`comparison.csv`) that is created from the union of the data outputted by the analysis application and the initial dataset.

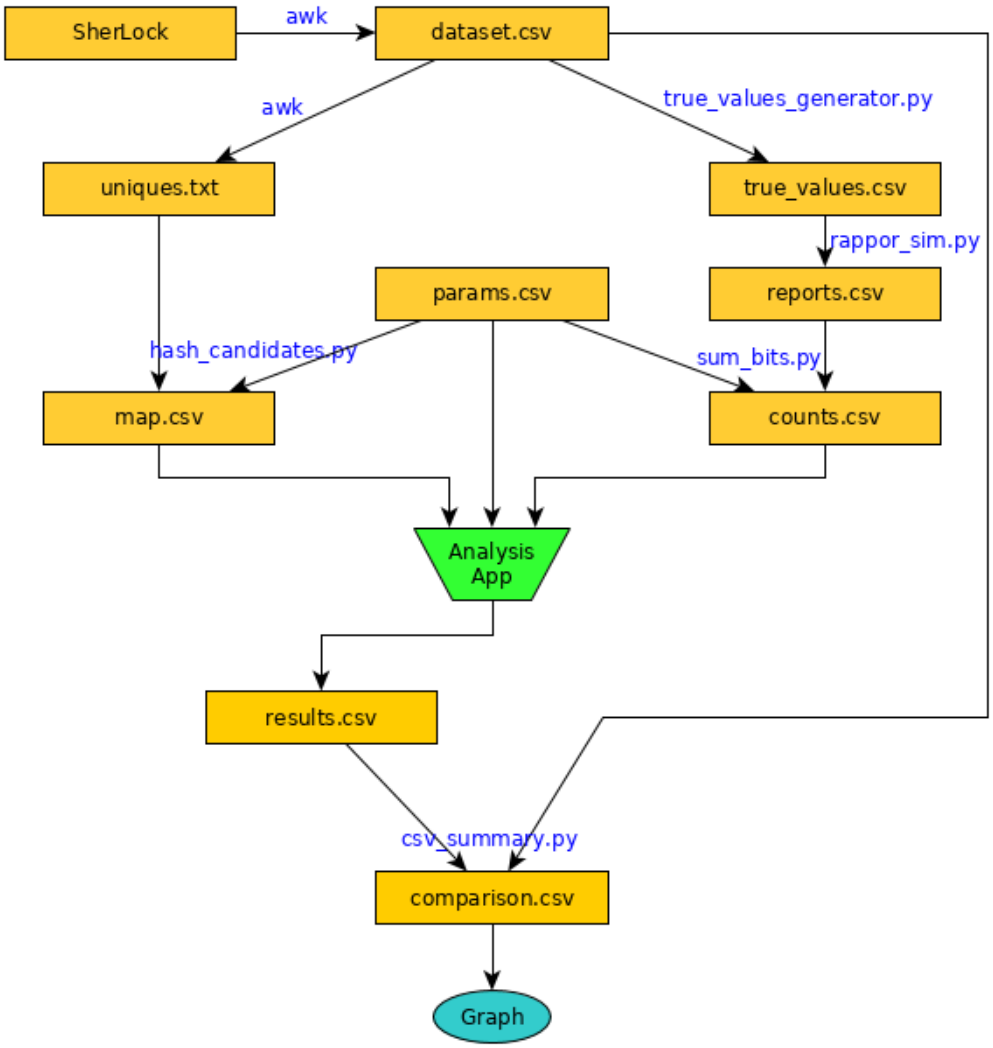

Figure 4.6: Design of the crafted pipeline required for the experiment

4.3.1 uniques.txt

A list of unique string values, one per line. All candidate strings must be in this list which, together with `params.csv`, can be used to generate `map.csv` through the script `hash_candidates.csv`.

4.3.2 true_values.csv

A file with a header "client,cohort,value" which will contain the real answers of the surveyed users in the following rows. The script `rappor_sim.py` can be used to create the RAPPOR reports from the original data, allowing a simpler experimentation process. An excerpt of a file with this format can be seen on Figure 4.7.

	A	B	C
1	client	cohort	value
2	c1	1	v11
3	c1	1	v13
4	c1	1	v13
5	c1	1	v1
6	c1	1	v16
7	c1	1	v2
8	c1	1	v14
9	c1	1	v9
10	c1	1	v2
11	c1	1	v6
12	c2	2	v11
13	c2	2	v11
14	c2	2	v10
15	c2	2	v44

Figure 4.7: File true_values.csv obtained from the demo

4.3.3 reports.csv

With the header "client,cohort,bloom,prr,irr", all the following rows contain that information for every report produced, with every stage in the production of the Bloom filters represented in binary form, as Figure 4.8 shows[2]. From this file, the `counts.csv` file can be generated through the script `sum_bits.py`.

[2]This file is shown from a text editor instead of in a spreadsheet because the binary format was interpreted by the spreadsheet software as an extremely big integer, converting the columns "bloom", "prr" and "irr" to an exponential representation.

```
client,cohort,bloom,prr,irr
c1,1,00000000000000000001000000000000,10001011100001100001100000100101,10101001100010001001100000101101
c1,1,00000000000000001000000000000000,00011011010000101010010000010100,01011011110100011001111010111101
c1,1,00000000000000001000000000000000,00011011010000101010010000010100,10010011010000000010100000010000
c1,1,00000000000000000000000000001000,00010010101010100010100001011010,00110010101011101010001101111011
c1,1,00000000000000000001000000000000,11011100100000000010000101100001,11001100000100001000000111100011
c1,1,00000000000000000000000000000001,00100100000000000100001000000100,00111101000010000100001000000100
c1,1,00000000000001000000000000000000,11000001000001000110001001001000,10011001000011100101001100001100
c1,1,00000000000000000000000000000001,00100100000000000100001000000100,00110100000001000100110000010100
c1,1,00000000000000000000000000000001,00100100000000000100001000000100,01101010100000000000011010011110
c1,1,00000000000000000000000100000000,00000010000011000000000100000010,00000010000011001110000000000010
c2,2,00000000000010000000000000000000,10011001001000110001000001000000,11011001011000110001000001001000
c2,2,00000000000010000000000000000000,10011001001000110001000001000000,10000101000010110000001101001011
c2,2,00000000000000000000000000100000,00000000000000000001010110110000,01000000000001000000010100111011
c2,2,10000000000000000000000000000000,10000000000011001100001100000110,00000001110011010010001110100011
```

Figure 4.8: File true_values.csv obtained from the demo

4.3.4 results.csv

The data inserted into the analysis web application outputs a CSV file with a header and columns that hold the detected candidate strings, the estimate count of appearances and a number of statistical measures about the recovered data, as Figure 4.9 show.

	A	B	C	D	E	F	G
1	string	estimate	std_error	proportion	prop_std_error	prop_low_95	prop_high_95
2	com.tranzmate	379	123	0.0379	0.0123	0.013792	0.062008
3	com.sec.android.app.music	303	141	0.0303	0.0141	0.002664	0.057936
4	com.sec.android.app.sns3	226	70	0.0226	0.007	0.00888	0.03632
5	com.samsung.android.app.headlines	216	79	0.0216	0.0079	0.006116	0.037084
6	com.sec.android.soagent	195	68	0.0195	0.0068	0.006172	0.032828
7	com.sec.android.app.mt	173	79	0.0173	0.0079	0.001816	0.032784

Figure 4.9: File results.csv obtained from the analysis web application

4.3.5 comparison.csv

This final file shows a column for the candidate strings, other for the count of appearances on the original dataset and other for the counts on the recovered data (Figure 4.10). This file is created by the script `csv_summary.py`, which can be seen on Appendix E.

	A	B	C
1	value	real	estimate
2	com.android.systemui	143378	143722
3	com.google.android.googlequicksearchbox	109777	107982
4	com.android.server.telecom	69184	66356
5	com.android.nfc	68419	68030
6	org.simalliance.openmobileapi.service	61492	61561
7	com.google.android.inputmethod.latin	54158	54155
8	com.google.android.gms	48778	48430
9	com.bgu.congeor.sherlockapp	45774	44436
10	com.samsung.SMT	33624	32135
11	com.samsung.android.fingerprint.service	31462	31545
12	com.sec.spp.push	27222	23769
13	com.facebook.katana	26944	28862
14	com.android.incallui	21133	20785

Figure 4.10: File comparsion.csv created by csv_summary.py

4.4 Dataset Creation

4.4.1 Data Extraction

The data needed for feeding the pipeline consists simply on a CSV file separated by commas, with a header and two columns, the first one with a unique identifier for every client, and the second one with the value to encode. Using the previously described dataset (named `Applications.csv`), the following AWK command was performed to obtain the final dataset:

```
awk 'BEGIN{FS=","}{print $2","$5;}' Applications.csv > dataset.csv
```

A manual inspection was required to remove the final blank line. To create the list of unique strings, the following command was issued:

```
cut -d ',' -f 5 Applications.csv | sort | uniq > uniques.txt
```

From this list, the before mentioned blank space and the header "Package-Name" had to be removed via manual inspection.

4.4.2 Subsampling

A script that creates subsamples from aleatory selections of reports was elaborated to create the datasets for the experiment (Appendix C). This script enables to create such subsamples choosing the amount of users and reports per user to select.

This way, the populations of 10 000 and 100 000 pieces of data were created by selecting just one report from users until completing the list, while the population of 1 200 000 pieces of data was constructed selecting 4 reports from 300 000 users.

4.5 Experiment Execution

4.5.1 Scenario Creation

The values of ϵ decided on the methodology were approximated using the calculator shown in Appendix A. As the number of hashes affects the recoverability conditions (Erlingsson et al., 2014), this value was left unchanged, setting it to 2, as it appears in 3 of the experiments performed by Erlingsson et al.. Table 4.1 show the configurations chosen for the experiment.

ϵ	**f**	**h**	**p**	**q**
0.1	0.75	2	0.5	0.55
1.0743	0.5	2	0.5	0.75
10.0184	0.01	2	0.05	0.9

Table 4.1: Values chosen to obtain the different ϵ for the experiment

Each of these sets of values were collected in three different files with the same structure as the already explained `params.csv`. The files were called `params_01.csv`, `params_1.csv` and `params_10.csv`.

4.5.2 Reports Generation

Once created the parameter files, the different series of report generation files were automated in a script that made use of the `pipeline.sh` script already

described (Appendix D). The automation script can be reviewed in Appendix G.

This script runs the pipeline for the three different datasets and the three different parameter groups, creating a total of 9 different scenarios.

4.5.3 Estimation Extraction

After the generation of the reports was finished, each set of files (`params.csv`, `counts.csv` and `map.csv`) was introduced into the analysis web application. The application offered a summary file with estimation data and a results file with the estimation of the appearance of the different candidate strings.

Graph Generation

The file `results.csv`, together with the original dataset for that concrete experiment, was used to generate the final comparison file through the script depicted in Appendix E. The resultant file compares the counts of candidate strings in the original distribution versus the estimation. Therefore, this file is suitable for being used on plotting software to obtain the final graphics. For the generation of the graphs in this work, the web service `https://plot.ly` was used. The graphs can be seen in Appendix I.

4.6 Conclusions

This chapter has represented the process for designing and impementing the tools to effect the experimentation phase after a conscientious examination of the available code and its structure.

During this analysis of the code, it was discovered that some differences were present between the description of the algorithm made by Erlingsson et al. and the implementation offered on GitHub:

First, the memoisation step described in literature was performed through a deterministic algorithm that could be better compared to a hashing operation than to a randomisation one. This fact changed the inner workings of the mechanism, but it was still a valid option for the PRR, as it satisfied the requirements of the memoisation phase. Moreover, this approach could subjectively be considered more secure, as it implies storing less information in the client.

Second, the size of the Bloom filters was limited to 32 bits in both the Python and the C++ clients. This exposed the fact that the code used for the experimentation phase in the paper that this work intended to compare to (Erlingsson et al., 2014) was necessarily different to the one offered in the GitHub repository. The Java client admitted bigger sizes of reports, but after several tests and attempts, the implemented modules did not offer acceptable results.

In spite of the difference on the Bloom filters' sizes, the experiment results should not be affected as long as enough hashes are provided to avoid excessive collisions, as the size of the filter should not alter the privacy levels of the test nor its recoverability properties (Erlingsson et al., 2014).

After backtracing the files required by the analysis application through the demonstrative pipeline, a custom stream of modules was tailored to perform the experiments. A considerable effort was put into automating these steps as much as possible, in an attempt to ease the repetition of the experiments, fostering possible future verifications by other researchers.

Before the execution of the experiment, the dataset was reshaped to satisfy the needs of the pipeline, and three subsamples of different sizes were extracted through a script that collected a defined number of responses from a certain amount of users to create the reduced datasets. The development of such script focussed on extracting all the data in the most aleatory manner possible.

Chapter 5

Evaluation

5.1 Introduction

In this chapter, the results obtained on the experimental phase will be analysed and compared. To establish an objective and precise evaluation methodology, the results will be compared by two criteria: Cases with different population size but identical ϵ values, and cases with different ϵ value on the same population. Critically analysing the scenarios by these two factors will help to evaluate more thoroughly the two research aims proposed on Chapter 3.

5.1.1 Data Presentation

The information collected from the experiment has been plotted in different graphics that can be seen on Appendix I. These figures will be referenced throughout the chapter while the obtained results are evaluated. In order to summarise the data extracted from the experiment, Table 5.1 is shown below, depicting the number of strings detected by the analysis app after these had been encoded in RAPPOR reports.

ϵ	**Pop. Size**	**Strings Present**	**Strings Detected**
0.1	10k	121	0
0.1	100k	140	1
0.1	1m	143	2
1	10k	121	7
1	100k	140	14
1	1m	143	47
10	10k	121	39
10	100k	140	60
10	1m	143	77

Table 5.1: Number of strings recovered

5.1.2 Data Analysis

In order to illustrate more objectively the results obtained, a Python script has been developed (Appendix H) to calculate the amount of detected strings that approximated the real count with a certain maximum margin of error. The percentages chosen for this comparison were 5, 10 and 20, the first two aimed to count precise approximations and the third one to discard values too distant to the original one. These numbers can be seen on Table 5.2.

ϵ	**Pop. Size**	**5%**	**10%**	**20%**
0.1	10k	0%	0%	0%
0.1	100k	0%	0%	0%
0.1	1m	0%	0%	0%
1	10k	14.3%	14.3%	14.3%
1	100k	14.3%	21.4%	50%
1	1m	27.6%	38.3%	51%
10	10k	18%	33.3%	59%
10	100k	33%	46.7%	66.7%
10	1m	53.24%	61%	77.9%

Table 5.2: Percentage of strings recovered according to different error margins

5.2 Same ϵ Values

The value of ϵ determines entirely the level of protection provided by differential privacy. According to McSherry and Mahajan (2010), an $\epsilon \leq 0.1$ should be considered as a strong guarantee of privacy, while an $\epsilon \geq 10$ should be considered as a weak guarantee. In this section, these called "strong" and "weak" levels will be tested with the previously generated datasets, to evaluate the quality of the recovered data that RAPPOR can offer in these situations. An intermediate level of $\epsilon = 1$, equidistant to the other two values by a proportion of 1:10, is also included to show the transition and appreciate better the changes that the recovered data suffer as the value of ϵ varies.

5.2.1 $\epsilon = 0.1$

Table 5.2 show how none of the recovered pieces of data from the smaller ϵ value could retrieve a single string with even a 20% margin of error. As it can be

seen on Figure 5.1 (a bigger version of all the figures discussed in this chapter can be seen on Appendix I), the biggest population at least registers correctly the most popular string. Nevertheless, in doing so, it also provides a considerably large false positive result in the tail of the distribution. This shows how, even being ten times higher than the smallest value found in literature (Sarwate, Monteleoni & Chaudhuri, 2009), this restriction is too constraining to be useful in, at least, datasets of the sizes used for this experiment. This also confirms the conclusions of McSherry and Mahajan (2010) saying that such a low value for ϵ produces considerably noisy responses, prone to offer a very high false positive rate.

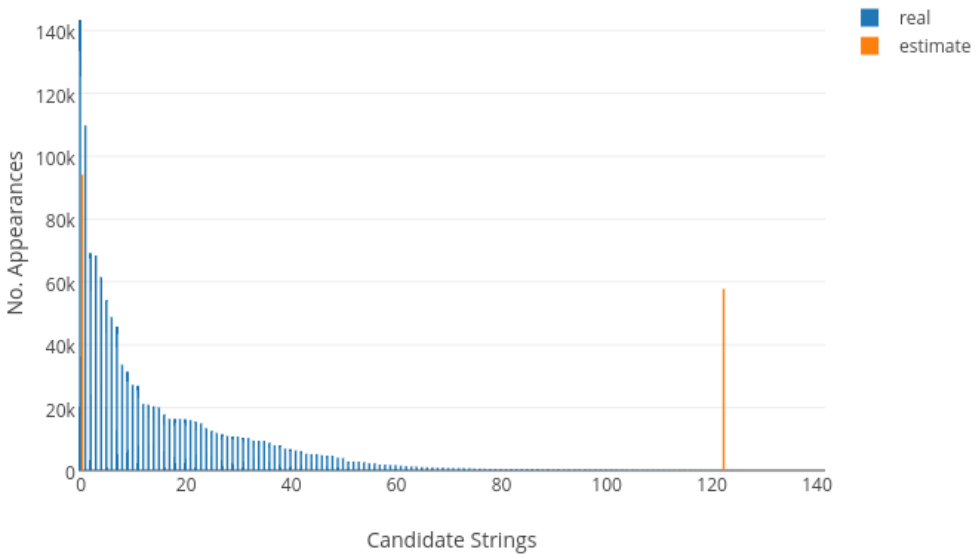

Figure 5.1: $\epsilon = 0.1$, population of 1 200 000 responses

In the case of Sarwate et al., it could be justified the usage of such a small value, as their project focusses on machine learning, a field where processing substantially larger datasets is rather common. Nevertheless, without further testing, it is not possible to determine whether the differential privacy concretely implemented by RAPPOR would successfully retrieve useful information in such conditions.

5.2.2 $\epsilon = 1$

Replicating the same privacy parameters used in the experiments of Erlingsson et al., Figures 5.2, 5.3 and 5.4 show how significant the size of the population is when choosing privacy values on RAPPOR.

The smallest population (Figure 5.2) barely detects some of the most popular strings, with a clear false positive in the tail of the distribution. Only 14.3% of the values are accurately approximated, as Table 5.1 show.

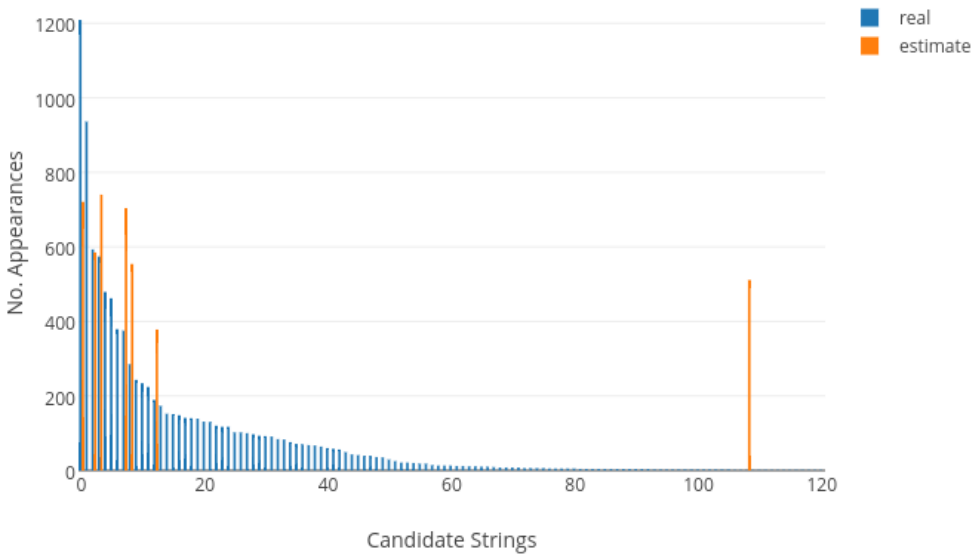

Figure 5.2: $\epsilon = 1$, population of 10 000 responses

In the intermediate population, shown on Figure 5.3, the most popular strings are properly detected to a certain level, with 50% of the detected values within the range of 20% error. The tail still shows some false positive results, but they are smaller in proportion. Despite the fact that the head of the estimated distribution seems to start shaping closer to the real data, only 14.3% of the detected strings were recovered with less than a 5% margin of error.

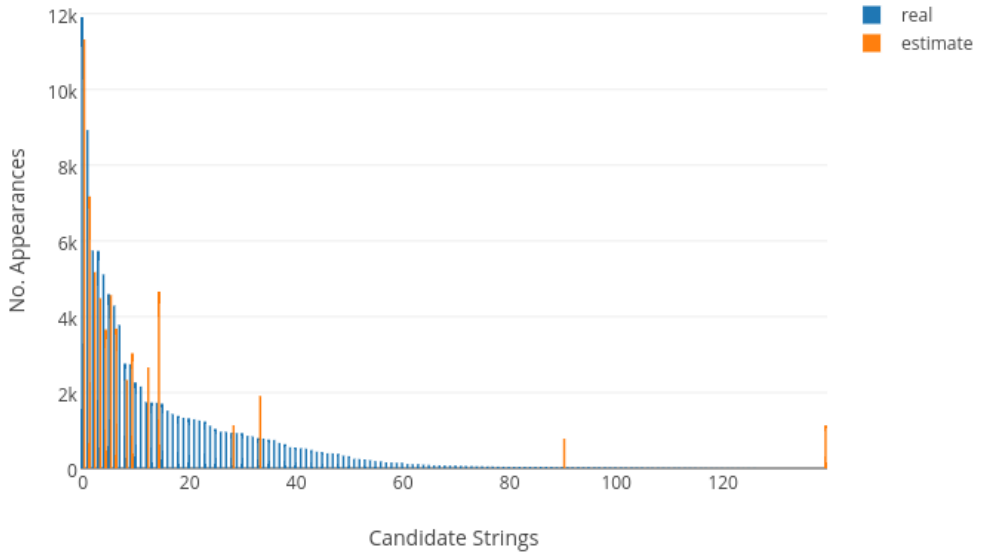

Figure 5.3: $\epsilon = 1$, population of 100 000 responses

Figure 5.4 shows how retrievability significantly increases given enough pieces of data. With ten times more data than the previous sample, the proportion of recovered strings almost doubles in the high accuracy range (5% margin of error). The proportion of strings detected on the 20% range stays steady, while the intermediate parameter grows from 21.4% to 38.3%. This situation, inverted from the one described in the transition from 10 000 to 100 000 users, could imply that, even when a minimum amount of reports is required to properly detect candidate strings, a higher increase in the amount of reports is largely beneficial to the accuracy of the estimations, but not as helpful in detecting new strings with an acceptable level of precision.

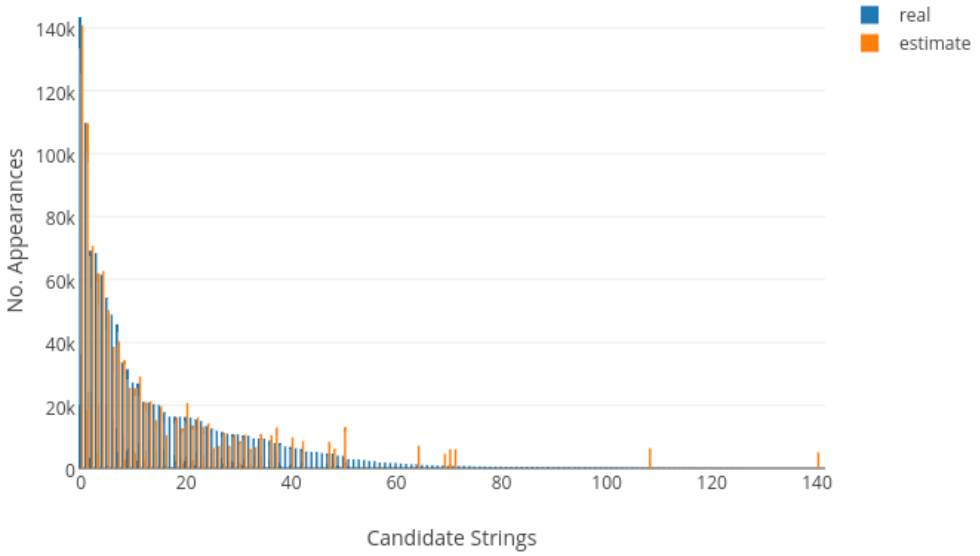

Figure 5.4: $\epsilon = 1$, population of 1 200 000 responses

5.2.3 $\epsilon = 10$

The data studied in this section comprise the weakest of the values of ϵ over the three populations. As Figure 5.5 shows, the general shape of the real distribution can already be appreciated on the recovered data, even with a relatively small population. It is worth highlighting that the proportion of strings detected with a 20% error is even bigger than in the 1 200 000 population with $\epsilon = 1$ (Table 5.2), while it decreases in five points in the 10% error limit, and almost ten in the 5% error limit, which could indicate that, even when the detection ratio can be highly improved by reducing the privacy guarantee, a high amount of data will still be needed to obtain precise data.

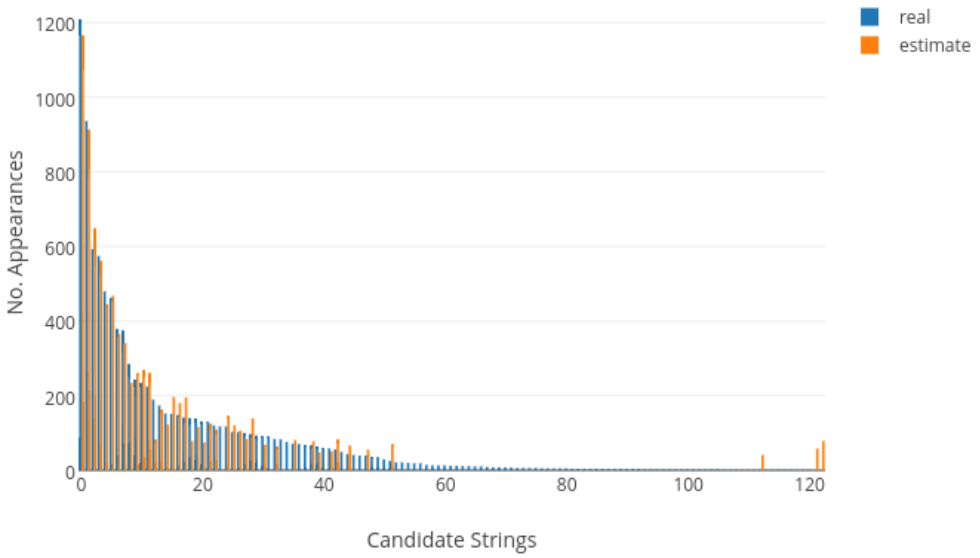

Figure 5.5: $\epsilon = 10$, *population of 10 000 responses*

In the mid-sized population, the recovered distribution already adopts very closely the shape of the original dataset, as Figure 5.6 shows. 66.7% of the recovered strings are inside the 20% margin of error, 46.7% are in the range of the 10% error and 33% reproduce the original count of such strings with more than 95% accuracy.

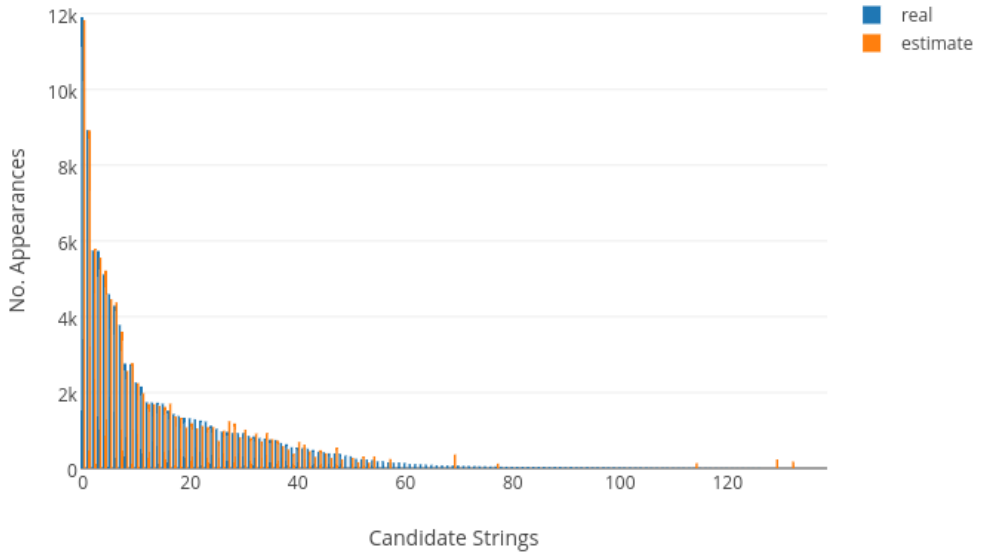

Figure 5.6: $\epsilon = 10$, population of 100 000 responses

The best case of retrieval of the original distribution can be seen on Figure 5.7, where the biggest population is processed with the lowest privacy guarantee. As it was expected from what was learnt from literature, the original and the estimated distributions match almost perfectly. The amount of detected strings has a small growth from the previous population size, compared to the strings detected with a 95% of precision, which could indicate that the strings with smaller appearances cannot be detected even with disproportionately sized datasets, while increasing the amount of reports collected contributes significantly to the accuracy of the strings detected.

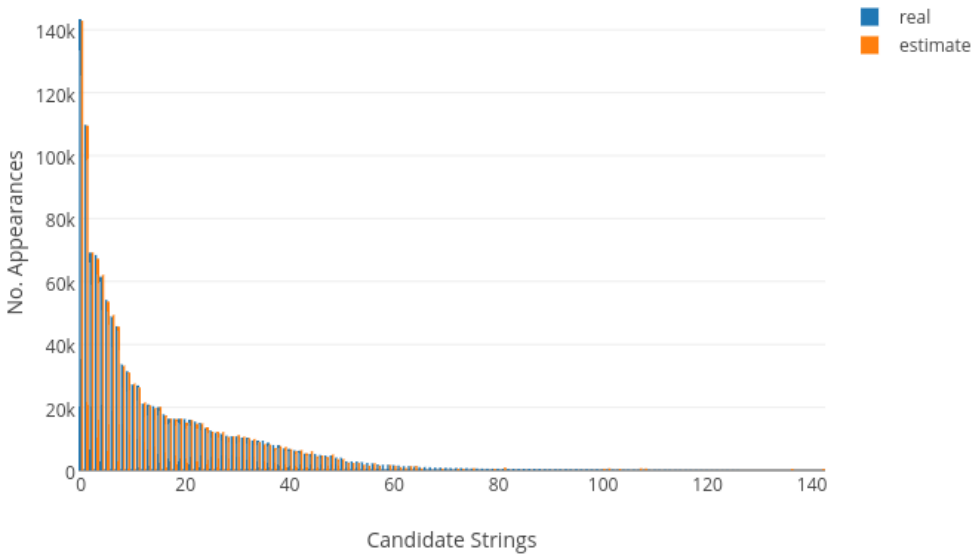

Figure 5.7: $\epsilon = 10$, population of 1 200 000 responses

5.3 Same Population Sizes

The samples exposed earlier on will be now grouped by population size, with the intention of evaluating the effects of altering ϵ on the same original distribution, and judge how this action affects to the retrievability of the data.

5.3.1 Population of 10 000 data

In the transition from $\epsilon = 0.1$ to $\epsilon = 1$, only 14.3% of the strings appear to be detected with precision, being the same figure present on the three levels of accuracy (Table 5.2). It is interesting to notice that, according to Figure 5.2, the most important strings in the original distribution would not be present in these groups, as their estimated value are considerably distant from the original. This could be due to the concentration of these values on a reduced number of cohorts, making the detection more difficult, given this level of privacy.

Passing from $\epsilon = 1$ to $\epsilon = 10$, while the proportion of acceptably enough detected strings (20% margin of error) grows considerably, the high precision classification suffers a small growth, in comparison: Even when the privacy guarantee is practically non-existent ($\epsilon = 10$), the improvement from the previous level is only of 3.7 points. This shows how even the smallest privacy protection requires a high number of reports to offer accurate results.

Looking at Figure 5.5, the original distribution is still far from being reconstructed by the estimation. As it was mentioned on Chapter 3, this population was composed of just one response per client, which could cause a more limited variety of data that would complicate the task of RAPPOR (Erlingsson et al., 2014).

5.3.2 Population of 100 000

A high growth of the 20% limit is shown on the first transition from 0.1 to 1 ϵ value, going from 0% to 50%. Almost half of the detected strings enter into the 10% error limit, and half of those reach the 95% accuracy. Figure 5.3 shows how the most accurate values concentrate around the head of the distribution, while the tail remains barely detected.

Changing ϵ to 10 causes the estimation to grow considerably closer to the original data (Figure 5.6), doubling the high precision detected value, with a more modest growth in the 20% error limit.

5.3.3 Population of 1 200 000

Even the biggest of the populations shows no successful results when being processed through the more restricting privacy parameter. Although the detection of the biggest string can be seen on Figure 5.1, not even this value is approximated with at least 80% accuracy.

The transition to $\epsilon = 1$ recreates the head of the distribution rather accurately (Figure 5.4), with 27.6% of the strings highly accurately detected.

The least demanding privacy restriction provides 77.9% of the strings detected with less than a 20% error. The biggest growth from $\epsilon = 1$ to $\epsilon = 10$ is appreciated in the high precision limits, offering a recovered distribution almost identical to the original one (Figure 5.7).

5.4 Conclusions

In this chapter, the data obtained from the experiments have been compared by different criteria, offering results that led to various deductions. The statements of Erlingsson et al. about the behaviour of RAPPOR to the variation of input parameters have been confirmed, and their experimental results have been verified. This fact strengthens the argument that the procedures followed in the experimentation process, both in the original work and in this thesis, are correct.

The privacy parameter that allowed to learn more from the retrievability of the data was $\epsilon = 1$, as this level of intermediate privacy guarantee allowed to observe a more progressive improvement on the fidelity of the recovered data as the population grew.

It has been seen how 10 000 responses have resulted to be insufficient for reconstructing an original distribution, unless extremely soft values of ϵ are employed. This event indicates that a minimum number of reports is necessary to recover reliable data, as RAPPOR requires a wide variety of responses to detect true positives. This dataset was built from a selection of users, choosing one string from each of them at random. This could affect the ability of RAPPOR to recover the data from the reports. Tests on different populations with the same size but different varieties of data could help to attain further understanding on the issue.

It becomes apparent that once the threshold of minimum reports required to obtain consistent results is reached, the fidelity of the recovered values is enhanced considerably, while the number of detected strings grows at a more moderated pace. This implies that, although a disproportionate collection of information could help to improve the precision of the recovered data, the number of detected strings would cease to grow eventually, delimiting the variety of information that can be learnt from the population.

A value of 0.1 ϵ-indistinguishability has demonstrated to be excessively strict for data in the range of millions or below. This value could possibly be sufficient for larger populations, although this statement remains uncertain without further experimentation.

For relaxed values of privacy guarantee, data is confidently retrieved even for small populations. Nevertheless, such values of ϵ should be employed with caution, as these parameters are considered weak and could expose sensitive data.

Chapter 6

Conclusions

6.1 Overall Conclusions

This project sought to evaluate the reliability of an implementation of differential privacy in providing protection to users when their data is collected, at the same time that it enabled data controllers to retrieve reliable information. In order to do so, an experimental methodology was developed to obtain different scenarios that allowed to measure the repercussions of altering the input values on the final results.

The reviewed technology is a relatively young project (2014) that is based on literature that defines privacy in a formal, measurable manner, and which has withstood testing and critique from other researchers without further discredit. No signs of adoption of the RAPPOR technology appear outside Google, which led to wonder why an open sourced project originated from solid theoretical work was not receiving further attention.

The empirical work performed on this dissertation proves that the project functions accordingly to its description and that its implementation is affordable even for small a development team. Providing additional information about the behaviour, recoverability properties and privacy guarantees that RAPPOR offers, the intention of this thesis is to foster further research on the field, as it has been proved that this technology, while being on its early stages, is based on strong foundations that ensure its quality. Moreover, it has been verified that the results obtained by Erlingsson et al. (2014) in their experiments are repeatable, ensuring the quality of their work.

6.2 Appraisal of Achievements

In this section, the original objectives will be reviewed in order to critically assess how the initial expectations have been met. As a reminder, the objectives are listed below:

1. Critical literature review of current research in the field of privacy preservation techniques. Special attention will be paid to their advantages, limitations and real world implementations.
2. Design of a rigorous testing methodology that allows to compare the effects of altering parameters (e.g., population size, privacy guarantee, etc.) in the final reports obtained in different scenarios generated from the combination of those input parameters, with the intention of correlating these variations to characteristics like retrievability and privacy protection levels.
3. Implementation of a piece of software that enables conducting the experiment. For this part of the thesis, the Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR), an implementation of differential privacy incorporated into the Chrome browser, will be used. The artifact will consist on a pipeline of modules that will take the original data through the whole process until generating the RAPPOR reports for their later analysis.
4. Discussion and evaluation of the obtained results, comparing them to the original data and judging the relationships between the different input and output metrics.

6.2.1 Literature Review

Covered on Chapter 2, the critical review of existent literature in privacy preserving mechanisms created a framework inside which the work made further on was better understood. Apart from noise addition, all the other classifications resulted to possess considerable disadvantages, and even randomisation can generate security problems when not properly implemented.

The thorough analysis of the formal definition of differential privacy and its critique enabled to verify the strength of the foundations over which RAPPOR was based, ensuring a high level of privacy protection by design. All critiques against this concept resulted to be disproved and based on conjectures or mis-

interpretations of the same works that were analysed in the literature review, placing differential privacy in a solid position that has remained unchanged over time.

The short literature on RAPPOR itself was closely studied, assessing its strong points and its limitations. This analysis enabled to understand the scenarios in which this technology can be used, offering a more complete understanding towards the design of the experimental phase.

6.2.2 Methodology

Along Chapter 3, the conditions under which the practical part of this work would be made were defined. Starting by extracting reflections from the literature review, it was decided that further study on the effects of varying conditions as the population size or the value of the privacy guarantee could help to better understand the properties and limitations of RAPPOR.

The parameters and data that would define the execution of the experiments were specified based on the studied literature. Furthermore, the criteria under which the results would be analysed further on was defined. This evaluation method, while not based on thorough statistical methods, aimed to be a simple yet objective and reliable method to judge two different properties of the recovered data: number of acceptable results and number of accurate results.

The main goal of the methodology design was to create a rigorous procedure that produced objective results that could be compared in order to discover occurrences that supported or refuted the predictions formulated from the analysis of the literature.

6.2.3 Implementation

Chapter 4 described the steps covered to design and implement the tools that would produce the experimental results over which the final evaluation of the thesis would be made. Analysing the code and the files used by a demonstration script in the repository, a tailored pipeline was built to create an experimenting process as automated as possible.

While designing such pipeline, the review of the code revealed that there was a limit of 32 bits to the size of the Bloom filters used on the RAPPOR reports. This issue implied that the experiments produced by Erlingsson et al. (2014) were

effectuated with a different implementation of their algorithm, as such experiments used bigger filters. A considerable amount of effort was put into trying to produce modules for the pipeline that accepted bigger Bloom filters without success, and the idea was finally dismissed due to the claim that the size of the Bloom filter should not affect the recoverability properties of the reports (Erlingsson et al., 2014).

After adapting the dataset to the necessities of the pipeline and generating the subsets of population, 9 diverse scenarios were obtained from the combination of three datasets and three values of the privacy guarantee.

The thorough documentation of all the steps covered, together with the design of auxiliary tools that help to automate the experimentation process, intends to ease the repetition and verification of the empirical work by future investigators interested in auditing the results obtained in this work.

6.2.4 Evaluation

In Chapter 5, all data obtained from the execution of the experiments were gathered and analysed. First of all, the parameters that would be observed on these comparisons were defined. The gross quantity of recovered strings did not allow an elaborated discussion, as it was not possible to judge the quality of the retrieved data. To remediate this limitation, it was decided to observe the amount of strings that could be recovered with different percentages of accuracy (80%, 90% and 95%).

This classification allowed to assess the variations on the recoverability of the experiment when the privacy guarantee or the population size changed by comparing each sample with the other samples that had one of the two parameters in common.

It was discovered that $\epsilon = 0.1$ was a privacy guarantee too high to be met even in the most optimistic of the scenarios, remaining uncertain whether this value would be usable for bigger populations. The 10 000 responses population only offered reliable results when the privacy guarantee was lowered to almost non-existent levels ($\epsilon = 10$).

It was appreciated that after certain threshold of population size, data started to be meaningful, observing great increases of accurately enough detected strings. After that initial growth, the rise of acceptable detected strings slowed, starting to increase the precision of the already discovered data instead. This lead to the conclusion that, from the aforementioned threshold onwards, collecting more

reports would increase the quality of the retrieved data, but not its diversity in such abundance.

The main priority in the production of this chapter was to ensure a thorough review of the results obtained in the experimental phase, also allowing a verification of the whole empirical process by comparing the outcomes with the ones obtained by Erlingsson et al. (2014). The results of both works are highly comparable, ensuring the correct procedure of the experimental phase.

6.3 Additional Experimentation: $0.5 \leq \epsilon \leq 1$

One of the conclusions that were reached during the experiments was that $\epsilon = 0.1$ was an excessively high privacy requirement for providing useful data. As $\epsilon = 1$ provided satisfactory results when used on the two bigger populations, it was decided to further investigate the behaviour of the recovered distributions in the space between the two values.

A new set of values (0.5, 0.6, 0.7, 0.8, 0.9) were defined for ϵ (Table 6.1) in order to study the evolution of the recoverability in the higher part of the interval described earlier. The lower half was left out of the scope of this new experiment, as early results with $\epsilon = 0.5$ already showed that the value offered a poor improvement from $\epsilon = 0.1$, as it will be seen. Discarding the lower half of the spectrum allowed to perform more experiments in the higher part, providing a higher resolution of the results.

ϵ	**f**	**h**	**p**	**q**
0.5	0.67	2	0.52	0.7
0.599	0.76	2	0.55	0.75
0.705	0.76	2	0.55	0.75
0.805	0.75	2	0.56	0.76
0.9	0.55	2	0.56	0.78

Table 6.1: Values chosen to obtain the different values of ϵ

Table 6.2 shows the results obtained on the three populations, where the total amount of detected strings are reported, together with the ones that approximated their count to the original ones in, at least, 80%. This time, the figures provided show the count of strings, instead of its proportion, as this representation was considered more appropriate to depict the graphs that will be discussed further on. The high precision results (5% and 10% tolerances) were not

considered, as the main aim of this additional experiment was to observe how the recoverability evolved, not its accuracy.

Pop. Size	ϵ	**Strings Detected**	**20% Tolerance**
10k	0.5	1	0
10k	0.6	2	0
10k	0.7	0	0
10k	0.8	3	0
10k	0.9	1	0
10k	1	7	1
100k	0.5	8	2
100k	0.6	11	3
100k	0.7	12	3
100k	0.8	16	4
100k	0.9	15	5
100k	1	14	7
1m	0.5	22	11
1m	0.6	26	15
1m	0.7	28	17
1m	0.8	33	17
1m	0.9	35	21
1m	1	47	24

Table 6.2: Strings recovered

The smallest population shows almost null results in this experiment, as it was expected from earlier results. The other two populations tended to grow as the privacy guarantee was slowly decreased.

On Figure 6.1, the population of 100 000 responses appears to grow linearly with an abrupt point at $\epsilon = 0.6$, which can hardly be considered relevant, given the reduced scale in which this experiment develops.

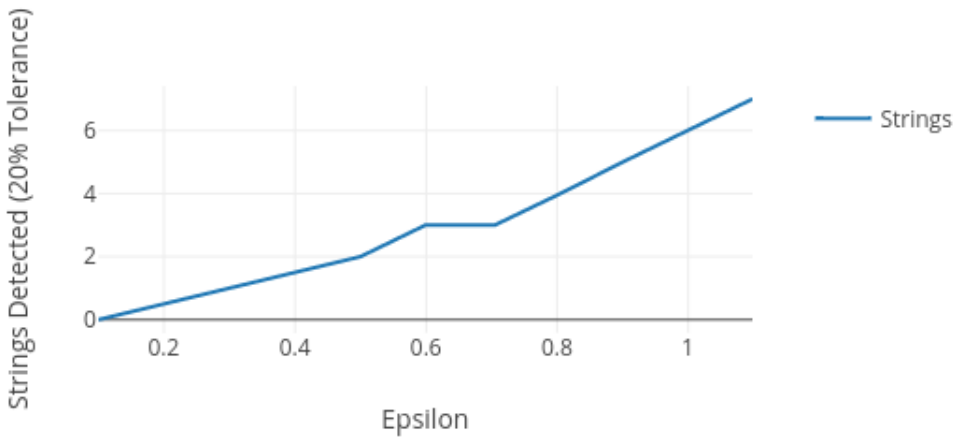

Figure 6.1: Evolution of the number of recovered strings with 20% tolerance in the population of 100 000 responses

Figure 6.2 shows a more accentuated growth of the number of successfully recovered strings. In spite of this, the population of 1 200 000 responses also appears to have a linear behaviour when the value of ϵ is gradually increased.

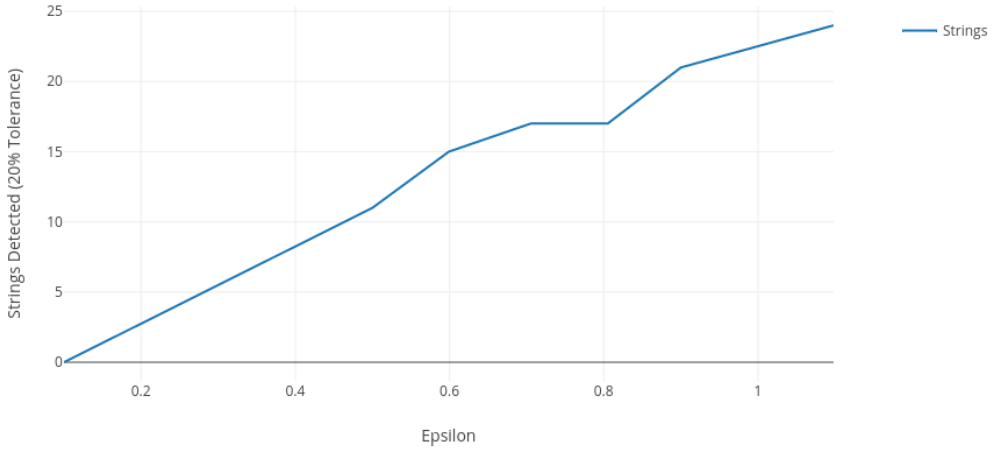

Figure 6.2: Evolution of the number of recovered strings with 20% tolerance in the population of 1 200 000 responses

The experiment showed how even for reduced increments of ϵ a significant improvement of the retrievability of the data can be seen, which indicates that the election of this parameter should not be taken lightly, as a small variation of it can translate into considerable changes in the fidelity of the recovered information.

Regarding the linear results of the two bigger populations, it can be deduced that such thing as an inflexion point from where the recoverability of the data grows exponentially should not exists, making the behaviour of the recovered information rather predictable.

In spite of these findings, the assumptions above made are only based on the study of a small area of the range of ϵ, and a more extensive analysis of its spectrum should be performed to assure such statements in a generalised manner.

6.4 Future Work

One of the biggest setbacks along the project was the impossibility of generating Bloom filters bigger than 32 bits. Even though the results from Erlingsson et al. have been successfully verified, this issue prevented from replicating the exact conditions of such empirical work. A drastic improvement to this project would be altering the code to make this feature available. This would help to create a more versatile tool, adaptable to different situations, and to completely replicate the conditions of the original experiments. Additional tests over different Bloom filters would also verify that the variation of the size of the filters does not affects the recoverability of the reports, as Erlingsson et al. assure.

The smallest of the chosen values of ϵ demonstrated to be impractical for populations of up to 1 200 000 reports. A possible additional check would be to determine if such a constraining privacy guarantee can be used with bigger populations, or if, on the contrary, this value makes RAPPOR to generate reports too noisy to be of any use.

Choosing three different sizes, the subsampled datasets were created by randomly selecting one report per user, in the case of the two smaller populations, and four different reports per user for the biggest one. Testing populations of the same size with different number of users and responses per user (e.g., 1 response for 1000 users, 2 responses for 500 users, 4 responses for 250 users, etc.) would help to further understand the influence of the characteristics of the population in the recoverability of the reports.

As it was previously discussed, the literature states that there exists an inflexion point where the bigger size of a dataset over another can be a disadvantage, as the less present strings become relatively smaller compared to the head of the distribution, causing them to be confused with noise when retrieving the data. This statement is not supported by empirical data of any kind, and it has not been noticed through the empirical phase of this work. A study of this phenomenon would help researchers or organisations willing to employ RAPPOR on their data collection operations to assess the target population size, optimising the recoverability of the data while protecting the privacy of the users.

Although the method employed to compare the different results aimed to offer unbiased and measurable data that enabled a thorough and objective discussion and evaluation of the experiments, it lacks a formal background based on statistics literature. A more thorough and verified method to compare the distributions obtained in the experimental phase could offer additional information that would lead to further discussion of the results.

References

Aggarwal, C. C. & Yu, P. S. (2008a). A General Survey of Privacy-Preserving Data Mining Models and Algorithms. In *Privacy-preserving data mining: Models and algorithms* (pp. 11–52). Springer US. Retrieved from http://link.springer.com/10.1007/978-0-387-70992-5{_}2 doi: 10.1007/978-0-387-70992-5_2

Aggarwal, C. C. & Yu, P. S. (2008b). A Survey of Randomization Methods for Privacy-Preserving Data Mining. In *Privacy-preserving data mining: Models and algorithms* (pp. 137–156). Springer US. Retrieved from http://link.springer.com/10.1007/978-0-387-70992-5{_}6 doi: 10.1007/978-0-387-70992-5_6

Article 29 Data Protection Working Party. (2014). *Opinion 05/2014 on Anonymisation Techniques* (Tech. Rep.). Retrieved from http://ec.europa.eu/justice/data-protection/index{_}en.htm

Bambauer, J., Muralidhar, K. & Sarathy, R. (2014). Fool's Gold: An Illustrated Critique of Differential Privacy. *Vanderblit Journal of Entertainment and Technology*, *16*(4), 701–755. Retrieved from http://www.jetlaw.org/wp-content/uploads/2014/06/Bambauer{_}Final.pdf

Barbaro, M. & Zeller, T. (2006). A Face Is Exposed for AOL Searcher No. 4417749. *New York Times*(4417749), 1–3.

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, *13*(7), 422–426.

Chromium Developers. (2015). *Rappor (Randomized Aggregatable Privacy Preserving Ordinal Responses) - The Chromium Projects*. Retrieved 2017-04-27, from https://www.chromium.org/developers/design-documents/rappor

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1–38.

Domingo-Ferrer, J. & Torra, V. (2008, mar). A critique of k-anonymity and some of its enhancements. In *3rd international conference on availability, security, and reliability, proceedings* (pp. 990–993). IEEE. Retrieved from http://ieeexplore.ieee.org/document/4529451/ doi: 10.1109/ARES.2008.97

Dwork, C. (2006). LNCS 4052 - Differential Privacy. *Automata, Languages and Programming*, *33*, 1–12. Retrieved from http://download.springer.com/static/pdf/868/chp{%}253A10.1007{%}252F11787006{_}1.pdf?originUrl=http{%}253A{%}252F{%}252Flink.springer

.com{%}252Fchapter{%}252F10.1007{%}252F11787006{_}1{&}token2=exp=1492852932{~}acl={%}252Fstatic{%}252Fpdf{%}252F868{%}252Fchp{%}2525253A1.1007{%}2525252F1178

Dwork, C. (2008). *Differential Privacy: A Survey of Results*. Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://link.springer.com/10.1007/978-3-540-79228-4{_}1 doi: 10.1007/978-3-540-79228-4_1

Dwork, C. (2011a). Deffinition of Differential Privacy. In *Encyclopedia of cryptography and security* (pp. 338–340). Boston, MA: Springer US. Retrieved from http://www.springerlink.com/index/10.1007/978-1-4419-5906-5{_}752 doi: 10.1007/978-1-4419-5906-5_752

Dwork, C. (2011b, jan). A firm foundation for private data analysis. *Communications of the ACM*, *54*(1), 86–95. Retrieved from http://portal.acm.org/citation.cfm?doid=1866739.1866758 doi: 10.1145/1866739.1866758

Dwork, C., Mcsherry, F., Nissim, K. & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd theory of cryptography conference (tcc)* (pp. 265–284). Retrieved from http://download.springer.com/static/pdf/997/chp{%}253A10.1007{%}252F11681878{_}14.pdf?originUrl=http{%}3A{%}2F{%}2Flink.springer.com{%}2Fchapter{%}2F10.1007{%}2F11681878{_}14{&}token2=exp=1492166185{~}acl={%}2Fstatic{%}2Fpdf{%}2F997{%}2Fchp{%}25253A10.1007{%}25252F11681878{_}14.pdf{%}3ForiginUr

Dwork, C. & Roth, A. (2013). The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, *9*(3-4), 211–407. Retrieved from http://www.nowpublishers.com/articles/foundations-and-trends-in-theoretical-computer-science/TCS-042 doi: 10.1561/0400000042

Dwork, C., Rothblum, G. N. & Vadhan, S. (2010, oct). Boosting and differential privacy. In *Annual ieee symposium on foundations of computer science, focs* (pp. 51–60). IEEE. Retrieved from http://ieeexplore.ieee.org/document/5670947/ doi: 10.1109/FOCS.2010.12

Dwork, C. & Smith, A. (2009). Differential Privacy for Statistic: What we Know and What we Want to Learn. *Journal of Privacy and Confidentiality*, *1*(2), 135–154. Retrieved from http://repository.cmu.edu/cgi/viewcontent.cgi?article=1008{&}context=jpc

El Emam, K., Jonker, E., Arbuckle, L. & Malin, B. (2011, dec). A Systematic Review of Re-Identification Attacks on Health Data. *PLoS ONE*, *6*(12). Retrieved from http://dx.plos.org/10.1371/journal.pone.0028071 doi: 10.1371/journal.pone.0028071

Erlingsson, Ú. (2014). *Research Blog: Learning Statistics with Privacy, aided by the Flip of a Coin.* Retrieved 2017-04-27, from https://research.googleblog.com/2014/10/learning-statistics-with-privacy-aided.html

Erlingsson, Ú. (2016, jun). Data-Driven Software Security: Models and Methods. In *2016 ieee 29th computer security foundations symposium (csf)* (pp. 9–15). IEEE. Retrieved from http://ieeexplore.ieee.org/document/7536362/ doi: 10.1109/CSF.2016.40

Erlingsson, Ú., Pihur, V. & Korolova, A. (2014). RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 acm sigsac conference on computer and communications security - ccs '14* (pp. 1054–1067). New York, New York, USA: ACM Press. doi: 10.1145/2660267.2660348

Fanti, G., Pihur, V. & Erlingsson, Ú. (2016, jan). Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries. *Proceedings on Privacy Enhancing Technologies*(3), 1–21. Retrieved from http://fanti.web.engr.illinois.edu/pets2016.pdfhttp://www.degruyter.com/view/j/popets.2016.2016.issue-3/popets-2016-0015/popets-2016-0015.xml doi: 10.1515/popets-2016-0015

Frank, M., Biedert, R., Ma, E., Martinovic, I. & Song, D. (2013, 1). Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *Information Forensics and Security, IEEE Transactions on*, *8*(1), 136 -148. doi: http://dx.doi.org/10.1109/TIFS.2012.2225048

Giakkoupis, G., Guerraoui, R., Jégou, A., Kermarrec, A.-M. & Mittal, N. (2015). Privacy-Conscious Information Diffusion in Social Networks. In Y. Moses (Ed.), *Proceedings of the 29th international symposium on distributed computing (disc)1* (pp. 480–496). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://download.springer.com/static/pdf/997/chp{%}253A10.1007{%}252F978-3-662-48653-5{_}32.pdf?originUrl=http{%}3A{%}2F{%}2Flink.springer.com{%}2Fchapter{%}2F10.1007{%}2F978-3-662-48653-5{_}32{&}token2=exp=1494940518{~}acl={%}2Fstatic{%}2Fpdf{%}2F997{%}2Fchp{%}25253A10.1007{%}25252F978-3-66 doi: 10.1007/978-3-662-48653-5

Goddyn, B. (2001). Defining anonymity and its dimensions in the electronic world. *Science*.

Goldwasser, S., Ishai, Y. & Nielsen, J. B. (2016). *Test-of-Time Award.* Retrieved 2017-06-13, from https://www.iacr.org/workshops/tcc/awards.html

Greenberg, A. (2016). *Apple's 'Differential Privacy' Is About Collect-*

ing Your Data—But Not Your Data. Retrieved 2017-04-27, from https://www.wired.com/2016/06/apples-differential-privacy-collecting-data/

Hachman, M. (2015). *The price of free: how Apple, Facebook, Microsoft and Google sell you to Advertisers.* Retrieved 2017-06-01, from http://www.pcworld.com/article/2986988/privacy/the-price-of-free-how-apple-facebook-microsoft-and-google-sell-you-to-advertisers.html

Harkiolakis, N. (2013). Right to Privacy. In *Encyclopedia of corporate social responsibility* (pp. 2082–2087). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://link.springer.com/10.1007/978-3-642-28036-8{_}453 doi: 10.1007/978-3-642-28036-8_453

Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B. C. & Roth, A. (2014, feb). Differential privacy: An economic method for choosing epsilon. In *Proceedings of the computer security foundations workshop* (Vol. 2014-Janua, pp. 398–410). Retrieved from http://arxiv.org/abs/1402.3329 doi: 10.1109/CSF.2014.35

Hu, X., Yuan, M., Yao, J., Deng, Y., Chen, L., Yang, Q., ... Zeng, J. (2015, aug). Differential privacy in telco big data platform. *Proceedings of the VLDB Endowment*, *8*(12), 1692–1703. Retrieved from http://dl.acm.org/citation.cfm?doid=2824032.2824067 doi: 10.14778/2824032.2824067

Huber, M., Müller-Quade, J. & Nilges, T. (2013). Defining privacy based on distributions of privacy breaches. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8260 LNCS*, 211–225. Retrieved from http://link.springer.com/10.1007/978-3-642-42001-6{_}15 doi: 10.1007/978-3-642-42001-6_15

Jiang, X., Zhao, Y., Wang, X., Malin, B., Wang, S., Ohno-Machado, L. & Tang, H. (2014). A community assessment of privacy preserving techniques for human genomes. *BMC medical informatics and decision making*, *14 Suppl 1*(Suppl 1), S1. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/25521230http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4290799 doi: 10.1186/1472-6947-14-S1-S1

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S. & Smith, A. (2011, jan). What Can We Learn Privately? *SIAM Journal on Computing*, *40*(3), 793–826. Retrieved from https://arxiv.org/pdf/0803.0924.pdfhttp://epubs.siam.org/doi/10.1137/090756090 doi: 10.1137/090756090

Kelly, D. (2016). *What are GDPR data controllers, processors, sub-*

jects and all the other actors? Retrieved 2017-06-08, from https://gdprchecklist.com/what-are-gdpr-data-controllers-processors-subjects-and-all-the-other-actors/

Kenthapadi, K., Mishra, N. & Nissim, K. (2005). Simulatable auditing. In *Proceedings of the twenty-fourth acm sigmod-sigact-sigart symposium on principles of database systems - pods '05* (p. 118). New York, New York, USA: ACM Press. Retrieved from http://portal.acm.org/citation.cfm?doid=1065167.1065183 doi: 10.1145/1065167.1065183

Kifer, D. & Machanavajjhala, A. (2011). No free lunch in data privacy. In *Proceedings of the 2011 international conference on management of data - sigmod '11* (p. 193). New York, New York, USA: ACM Press. Retrieved from http://portal.acm.org/citation.cfm?doid=1989323.1989345 doi: 10.1145/1989323.1989345

Kleinberg, J., Papadimitriou, C. & Raghavan, P. (2003, feb). Auditing Boolean attributes. *Journal of Computer and System Sciences*, *66*(1), 244–253. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0022000002000363 doi: 10.1016/S0022-0000(02)00036-3

Kosner, A. W. (2013). *New Facebook Policies Sell Your Face And Whatever It Infers.* Retrieved 2017-06-01, from https://www.forbes.com/sites/anthonykosner/2013/08/31/new-facebook-policies-sell-your-face-and-whatever-it-infers/{#}32fb210a273a

Li, N., Li, T. & Venkatasubramanian, S. (2007). t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *2007 ieee 23rd international conference on data engineering* (pp. 106–115). IEEE. Retrieved from http://www.utdallas.edu/{~}mxk055100/courses/privacy08f{_}files/tcloseness.pdfhttp://ieeexplore.ieee.org/document/4221659/ doi: 10.1109/ICDE.2007.367856

Lord, N. (2012). *How Mobile Apps are Invading Your Privacy Infographic.* Retrieved 2017-07-16, from www.veracode.com/blog/2012/05/how-mobile-apps-are-invading-your-privacy-infographic

Machanavajjhala, A., Kifer, D., Abowd, J., Gehrke, J. & Vilhuber, L. (2008, apr). Privacy: Theory meets Practice on the Map. In *2008 ieee 24th international conference on data engineering* (pp. 277–286). IEEE. Retrieved from http://ieeexplore.ieee.org/document/4497436/ doi: 10.1109/ICDE.2008.4497436

Machanavajjhala, A., Kifer, D., Gehrke, J. & Venkitasubramaniam, M. (2007, mar). l-diversity: Privacy Beyond k-Anonymity. *ACM Transactions on Knowledge Discovery from Data*, *1*(1), 3–es. Retrieved from https://pdfs.semanticscholar.org/004b/439ff1e6a15deedc7a7c4c6685f5ceafd237.pdfhttp://portal.acm.org/citation.cfm?doid=1217299.1217302 doi:

10.1145/1217299.1217302

McSherry, F. (2009). Privacy integrated queries. In *Proceedings of the 35th sigmod international conference on management of data - sigmod '09* (p. 19). Retrieved from https://www.microsoft.com/en-us/research/project/privacy-integrated-queries-pinq/ http://portal.acm.org/citation.cfm?doid=1559845.1559850 doi: 10.1145/1559845.1559850

McSherry, F. (2016). *Differential privacy for dummies*. Retrieved 2017-05-19, from https://github.com/frankmcsherry/blog/blob/master/posts/2016-02-03.md

McSherry, F. & Mahajan, R. (2010, August). Differentially-private network trace analysis. *SIGCOMM Comput. Commun. Rev.*, *40*(4), 123–134. Retrieved from http://doi.acm.org/10.1145/1851275.1851199 doi: 10.1145/1851275.1851199

McSherry, F. & Talwar, K. (2007, oct). Mechanism Design via Differential Privacy. In *48th annual ieee symposium on foundations of computer science (focs'07)* (pp. 94–103). IEEE. Retrieved from http://ieeexplore.ieee.org/document/4389483/ doi: 10.1109/FOCS.2007.66

Mirsky, Y., Shabtai, A., Rokach, L., Shapira, B. & Elovici, Y. (2016). SherLock vs Moriarty: A Smartphone Dataset for Cybersecurity Research. In *Proceedings of the 2016 acm workshop on artificial intelligence and security - aisec '16* (pp. 1–12). New York, New York, USA: ACM Press. Retrieved from http://dl.acm.org/citation.cfm?doid=2996758.2996764 doi: 10.1145/2996758.2996764

Mohammed, N., Chen, R., Fung, B. C. & Yu, P. S. (2011). Differentially private data release for data mining. In *Proceedings of the 17th acm sigkdd international conference on knowledge discovery and data mining - kdd '11* (p. 493). New York, New York, USA: ACM Press. Retrieved from http://dl.acm.org/citation.cfm?doid=2020408.2020487 doi: 10.1145/2020408.2020487

Morran, C. (2016). *Facebook Is Now Selling Your Web-Browsing Data To Adverstisers*. Retrieved 2017-06-01, from https://consumerist.com/2014/06/12/facebook-is-now-selling-your-web-browsing-data-to-advertisers/

Nabar, S. U., Kenthapadi, K., Mishra, N. & Motwani, R. (2008). A Survey of Query Auditing Techniques for Data Privacy. In *Privacy-preserving data mining: Models and algorithms* (pp. 415–431). Springer US. Retrieved from http://link.springer.com/10.1007/978-0-387-70992-5{_}17 doi: 10.1007/978-0-387-70992-5_17

Narayanan, A. & Shmatikov, V. (2006, oct). How To Break Anonymity of the Netflix Prize Dataset. Retrieved from http://arxiv.org/abs/cs/0610105

Narayanan, A. & Shmatikov, V. (2008, may). Robust De-anonymization of Large Sparse Datasets. In *2008 ieee symposium on security and privacy (sp 2008)* (pp. 111–125). IEEE. Retrieved from http://ieeexplore.ieee.org/document/4531148/https://www.cs.utexas.edu/{~}shmat/shmat{_}oak08netflix.pdf doi: 10.1109/SP.2008.33

Narayanan, A. & Shmatikov, V. (2010, jun). Myths and fallacies of "personally identifiable information". *Communications of the ACM*, *53*(6), 24. Retrieved from http://portal.acm.org/citation.cfm?doid=1743546.1743558 doi: 10.1145/1743546.1743558

Nguyên, T. T., Xiao, X., Yang, Y., Hui, S. C., Shin, H. & Shin, J. (2016, jun). Collecting and Analyzing Data from Smart Device Users with Local Differential Privacy. Retrieved from https://arxiv.org/pdf/1606.05053.pdf

Pfitzmann, A. & Köhntopp, M. (2001). Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology. In H. Federrath (Ed.), *Designing privacy enhancing technologies: International workshop on design issues in anonymity and unobservability berkeley, ca, usa, july 25–26, 2000 proceedings* (pp. 1–9). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-44702-4{_}1 doi: 10.1007/3-540-44702-4_1

Rossi, B. (2016). *Will differential privacy take favour in the enterprise?* Retrieved 2017-04-27, from http://www.information-age.com/will-differential-privacy-take-favour-enterprise-123461324/

Sarathy, R. & Muralidhar, K. (2009). Differential Privacy for Numeric Data. In *Joint unece/eurostat work session on statistical data confidentiality*. Bilbao, Spain. Retrieved from http://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.46/2009/wp.9.e.pdf

Sarwate, A., Monteleoni, C. & Chaudhuri, K. (2009). *Differentially private support vector machines* (Tech. Rep.).

Sweeney, L. (2000). Simple Demographics Often Identify People Uniquely. *Data Privacy Working Paper*, *3*. Retrieved from https://dataprivacylab.org/projects/identifiability/paper1.pdf

Sweeney, L. (2002, oct). k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*(05), 557–570. Retrieved from http://www.worldscientific.com/doi/abs/10.1142/S0218488502001648 doi: 10.1142/S0218488502001648

Wang, Y., Wu, X. & Hu, D. (2016). Using Randomized Response for Differential Privacy Preserving Data Collection. In *9th international workshop on privacy and anonymity in the information society (pais)*. Retrieved from http://ceur-ws.org/Vol-1558/paper35.pdf

Warner, S. L. (1965, mar). Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *Journal of the American Statistical Association*, *60*(309), 63–69. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/01621459.1965.10480775 doi: 10.1080/01621459.1965.10480775

Westrick, T. (2016). *Apple vs. Google vs. Microsoft: which company handles your data better?* Retrieved 2017-06-01, from https://decentralize.today/apple-vs-google-vs-microsoft-which-company-handles-your-data-better-a7022bd452b1

Zumel, N. (2015). *A Simpler Explanation of Differential Privacy*. Retrieved 2017-04-27, from http://www.win-vector.com/blog/2015/10/a-simpler-explanation-of-differential-privacy/

Appendix A

ϵ Calculator

```
#!/usr/bin/env python
import math

f = float(input("f: "))
h = float(input("h: "))
p = float(input("p: "))
q = float(input("q: "))

eInf=2*h*math.log((1-f/2)/(f/2))

qStar=(f/2)*(p+q)+(1-f)*q
pStar=(f/2)*(p+q)+(1-f)*p

eOne=h*math.log((qStar*(1-pStar))/(pStar*(1-qStar)))
print ("ln(3) = ", math.log(3))

print ("eInf = ", eInf)
print ("eOne = ", eOne)
```

Appendix B

True Values Generator

```
#!/usr/bin/python

import csv
import sys

"""Associates every client with a cohort, assumes that the dataset
    is ordered by clients.
Input: m,dataset.csv (client,value)
Output: true_values.csv (client,cohort,value))
"""

if len(sys.argv) != 3:
    print ("Usage: true_values_generator.py num_cohorts
        path_to_dataset.csv")
    sys.exit(1)

m = int(sys.argv[1])
filename = sys.argv[2]

ifile = open(filename, "rb")
reader = csv.reader(ifile)
writer = csv.writer(sys.stdout)
rownum = 0
client = ""
cohort = -1
for row in reader:
    # Save header row.
    if rownum == 0:
        writer.writerow(["client","cohort","value"])
    else:
        if client != row[0]:
            client = row[0]
            if cohort == -1 or cohort == m-1:
                cohort = 0
            else:
```

```
                cohort += 1
        newrow=[row[0],cohort,row[1]]
        writer.writerow(newrow)

    rownum += 1

ifile.close()
```

Appendix C

Subsampling Script

```
#!/usr/bin/python

"""Creates a subsample of the dataset, allowing to choose amount
    of users and reports per user.
Input: nusers, nreps, dataset.csv
Output: trimmed_dataset.csv (stdout)
"""

import csv
import sys
import linecache
import random

def indexUsers(reader):
    """Returns an index of users and the minimum amount of reports
        per user"""
    rowNum = 0
    minReps=sys.maxint
    tmpMinReps=sys.maxint
    client = ""
    listClients = []
    listIterator = -1
    firstRow = []
    for row in reader:
        if rowNum != 0:
            if client != row[0]:
                client = row[0]
                listClients.append(rowNum+1)

                if tmpMinReps < minReps:
                    minReps = tmpMinReps
                tmpMinReps = 1

            else:
                tmpMinReps += 1
```

```
        else:
            firstRow=row

        rowNum += 1

    return listClients, minReps, firstRow

if len(sys.argv) != 4:
    print ("Usage: dataset_trimmer.py nusers nreps dataset.csv")
    sys.exit(1)

inputNumUsers = int(sys.argv[1])
inputNumReps = int(sys.argv[2])
filename = sys.argv[3]

ifile = open(filename, "rb")
reader = csv.reader(ifile)

listUsers, minReps, firstRow = indexUsers(reader)

ifile.close()

writer = csv.writer(sys.stdout)


if  len(listUsers) < inputNumUsers:
    print "Insufficient amount of users"
    exit(1)

if minReps < inputNumReps:
    print "Insufficient minimum amount of reports per user"
    exit(1)

writer.writerow(firstRow)
for i in range(inputNumUsers):
    userPosition = listUsers.pop(random.randint(0,len(listUsers)-1))
    listOfReports =[]
    for j in range(minReps):
        listOfReports.append(linecache.getline(filename,
            userPosition+j).rstrip().split(','))
```

```
    for j in range(inputNumReps):
        writer.writerow(
            listOfReports.pop(
                random.randint(
                    0,len(listOfReports)-1
                    )
                )
            )

exit(0)
```

Appendix D

Pipeline

```
#!/bin/bash

if [[ $# -ne 3 ]]; then
    echo "Usage: $0 path_to_params.csv path_to_dataset.csv
       path_to_uniques.txt"
    exit 1
fi

FILESOK=true

if [[ ! -f $1 ]]; then
    echo "File $1 does not exist"
    FILESOK=false
fi

if [[ ! -f $2 ]]; then
    echo "File $2 does not exist"
    FILESOK=false
fi

if [[ ! -f $3 ]]; then
    echo "File $3 does not exist"
    FILESOK=false
fi

if [[ $FILESOK = false ]]; then
    exit 1
fi

readonly THIS_DIR=$(dirname $0)
readonly TMPDIR=tmpdir
export PYTHONPATH=$THIS_DIR/client/python
readonly PARAMS=$1
readonly DATASET=$2
readonly UNIQUES=$3
```

```
echo '[*] Extracting values from params.csv'
k=$(awk 'BEGIN{FS=","}{print $1;}' $PARAMS | tail -1)
h=$(awk 'BEGIN{FS=","}{print $2;}' $PARAMS | tail -1)
m=$(awk 'BEGIN{FS=","}{print $3;}' $PARAMS | tail -1)
p=$(awk 'BEGIN{FS=","}{print $4;}' $PARAMS | tail -1)
q=$(awk 'BEGIN{FS=","}{print $5;}' $PARAMS | tail -1)
f=$(awk 'BEGIN{FS=","}{print $6;}' $PARAMS | tail -1)
echo "k = $k"
echo "h = $h"
echo "m = $m"
echo "p = $p"
echo "q = $q"
echo "f = $f"

echo '[*] Deleting tmpdir'
rm -rf $TMPDIR

echo '[*] Creating tmpdir'
mkdir $TMPDIR

echo '[*] Creating map.csv'
bin/hash_candidates.py \
    $PARAMS \
    < $UNIQUES \
    > $TMPDIR/map.csv

echo '[*] Creating true_values.csv'
bin/true_values_generator.py \
    $m \
    $DATASET \
    > $TMPDIR/true_values.csv

echo '[*] Creating reports.csv'
time tests/rappor_sim.py \
    --num-bits $k \
    --num-hashes $h \
    --num-cohorts $m \
    -p $p \
    -q $q \
    -f $f \
    < $TMPDIR/true_values.csv \
    > $TMPDIR/reports.csv
```

```
echo '[*] Creating counts.csv'
bin/sum_bits.py \
    $PARAMS \
    < $TMPDIR/reports.csv \
    > $TMPDIR/counts.csv
```

Appendix E

Summary Generator

```
#!/usr/bin/env python2

"""Creates a CSV file to compare estimated and real results.
Input: dataset.csv, recovered.csv
Output: comparison.csv (stdout)
"""

import csv
import sys

if len(sys.argv) != 3:
    print ("Usage: csv_summary.py dataset.csv recovered.csv >
        comparison.csv")
    sys.exit(1)

dataset = sys.argv[1]
results = sys.argv[2]


ifile = open(results, "rb")
readerResults = csv.reader(ifile)
dictionary = dict()
rownum = 0

for row in readerResults:
    if rownum != 0:
        stringValue = row[0].strip('"')
        estimate = int(row[1])
        dictionary[stringValue] = [0,estimate]
    rownum +=1

ifile.close()

ifile = open(dataset, "rb")
readerDataset = csv.reader(ifile)
```

```
rownum = 0

for row in readerDataset:
    if rownum != 0:
        stringValue = row[1]
        if stringValue in dictionary:
            dictionary[stringValue][0] += 1
        else:
            dictionary[stringValue] = [1,0]
    rownum +=1

ifile.close()

writer = csv.writer(sys.stdout)
writer.writerow(["value","real","estimate"])
for item in dictionary:
    writer.writerow([item,dictionary[item][0],dictionary[item][1]])
```

Appendix F

Java Modules

F.1 Reports Generation

```
package javademo;

import java.nio.charset.StandardCharsets;
import com.opencsv.CSVReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.UUID;

public class Reports_generator {

   public static String printByteArray(byte[] b, int length) {
      String s = "";
      String total = "";
      for (int i = 0; i < length; i++) {
         s = ("0000000" + Integer.toBinaryString(0xFF &
             b[i])).replaceAll(".*(.{8})$", "$1");
         total += s;

      }

      return total;
   }

   public static void main(String[] args) throws IOException {

      if (args.length != 7) {
         System.err.println("Usage: reports_generator.jar k h m p q
             f input.csv > output.csv");
         System.exit(1);
      }

      String encoderId = "";
```

```
int numBits = Integer.parseInt(args[0]);
int numBloomHashes = Integer.parseInt(args[1]);
int numCohorts = Integer.parseInt(args[2]);
double probabilityP = Double.parseDouble(args[3]);
double probabilityQ = Double.parseDouble(args[4]);
double probabilityF = Double.parseDouble(args[5]);
String inputFile = args[6];

CSVReader reader = new CSVReader(new FileReader(inputFile));

String[] nextLine;
Encoder bloom = null;
Encoder prr = null;
Encoder irr = null;

int numrow = 0;
String[] row;
String currentClient = "";
byte[] userSecret;
while ((nextLine = reader.readNext()) != null) {
   if (nextLine != null) {
      if (numrow == 0) {
         row = new String[]{"client", "cohort", "bloom",
             "prr", "irr"};
      } else {
         String client = nextLine[0];
         int cohort = Integer.parseInt(nextLine[1]);
         String value = nextLine[2];

         if (!currentClient.equals(client)) {
            currentClient = client;

            userSecret = (UUID.randomUUID().toString() +
                UUID.randomUUID().toString())
               .getBytes(StandardCharsets.UTF_8);

            bloom = new Encoder(userSecret,
               client,
               numBits,
               0,
               0,
               1,
               numCohorts,
```

```
                cohort,
                numBloomHashes);

            prr = new Encoder(userSecret,
                client,
                numBits,
                probabilityF,
                0,
                1,
                numCohorts,
                cohort,
                numBloomHashes);

            irr = new Encoder(userSecret,
                client,
                numBits,
                probabilityF,
                probabilityP,
                probabilityQ,
                numCohorts,
                cohort,
                numBloomHashes);
        }
        row = new String[]{
            client,
            String.valueOf(irr.getCohort()),
            printByteArray(bloom.encodeString(value), numBits
                / 8),
            printByteArray(prr.encodeString(value), numBits /
                8),
            printByteArray(irr.encodeString(value), numBits /
                8)};

    }
    String s = "";
    for (int i = 0; i < row.length; i++) {
        s += row[i];
        if (i < (row.length - 1)) {
            s += ",";
        }
    }
    System.out.println(s);
    numrow++;
```

```
        }

     }

  }

}
```

F.2 Hashes Map Generator

```
package javademo;

import java.nio.charset.StandardCharsets;
import com.opencsv.CSVReader;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.UUID;

public class Map_generator {
   public static String printByteArray(byte[] b, int length) {
      String s = "";
      String total = "";
      for (int i = 0; i < length; i++) {
         s = ("0000000" + Integer.toBinaryString(0xFF &
             b[i])).replaceAll(".*(.{8})$", "$1");
         total += s;

      }
      return total;
   }

   public static String getBitsPosition(String bloom, int
       numHashes) {
      String positions = "";

      for (int j = 0; j < bloom.length(); j++) {
         if (bloom.charAt(j) == '1') {
            positions += "," + Integer.toString(j + 1);
```

```
      }
    }
    return positions;
  }

  public static void main(String[] args) throws IOException {
    if (args.length != 4) {
      System.err.println("Usage: map_generator.jar k h m
          input.csv > output.csv");
      System.exit(1);
    }
    String encoderId = "";
    int numBits = Integer.parseInt(args[0]);
    int numBloomHashes = Integer.parseInt(args[1]);
    int numCohorts = Integer.parseInt(args[2]);
    String inputFile = args[3];

    Encoder[] e = new Encoder[numCohorts];

    for (int i = 0; i < numCohorts; i++) {
      e[i] = new Encoder(
        (UUID.randomUUID().toString() +
            UUID.randomUUID().toString())
          .getBytes(StandardCharsets.UTF_8),
        "client",
        numBits,
        0,
        0,
        1,
        numCohorts,
        i,
        numBloomHashes
      );
    }

    BufferedReader br = new BufferedReader(new
        FileReader(inputFile));

    for (String nextLine = br.readLine(); nextLine != null;
        nextLine = br.readLine()) {
      String row = nextLine;

      for (int i = 0; i < numCohorts; i++) {
```

```
            String bloomFilter =
                printByteArray(e[i].encodeString(nextLine), numBits
                / 8);
            String positions = getBitsPosition(bloomFilter,
                numBloomHashes);
            row += positions;
          }
          System.out.println(row);

      }
      br.close();
    }

}
```

Appendix G

Experiment Script

```
#!/bin/bash

if [[ $# -ne 6 ]]; then
    echo "Usage: $0 params_01.csv params_1.csv params_10.csv
        dataset_10k.csv dataset_100k.csv dataset_1m.csv"
    exit 1
fi

#For directory creation, the files must have the above naming
    convention

readonly THIS_DIR=$(dirname $0)
readonly EXPERIMENT_DIR=$THIS_DIR/experiment
readonly PARAMS=($1 $2 $3)
readonly DATASETS=($4 $5 $6)
readonly TMPDIR=tmpdir

echo '[*] Deleting $EXPERIMENT_DIR'
rm -rf $EXPERIMENT_DIR

echo '[*] Creating $EXPERIMENT_DIR'
mkdir $EXPERIMENT_DIR

for PARAM in ${PARAMS[@]}; do
    for DATASET in ${DATASETS[@]}; do
        DIRPARAM=$(echo $PARAM | cut -d '_' -f 2 | cut -d '.' -f 1)
        DIRDATASET=$(echo $DATASET | cut -d '_' -f 2 | cut -d '.'
            -f 1)
        DIR=$EXPERIMENT_DIR/$DIRPARAM/$DIRDATASET
        $THIS_DIR/pipeline.sh $PARAM $DATASET uniques.txt
        mkdir -p $DIR
        mv $TMPDIR/counts.csv $TMPDIR/map.csv $DIR/
    done
done
```

```
exit 0
```

Appendix H

Statistics Collector

```
#!/usr/bin/env python2

"""Creates a CSV file that offers statistics about the results, as
    the amount of strings found with a determined margin of error
    (delim1, delim2 and delim3).
Input: comparison.csv delim1 delim2 delim3
Output: statistics.csv (stdout)
"""

import csv
import sys

if len(sys.argv) != 5:
    print ("Usage: statistics.py comparison.csv delim1 delim2
        delim3 > statistics.csv")
    print ("delim1, delim2 and delim3 should be numbers between 0
        and 100 and it should be the maximum percent threshold
        allowed to sum the amount of found strings.")
    sys.exit(1)

comparison = sys.argv[1]
delim1 = int(sys.argv[2])
delim2 = int(sys.argv[3])
delim3 = int(sys.argv[4])


ifile = open(comparison, "rb")
reader = csv.reader(ifile)
writer = csv.writer(sys.stdout)
rownum = 0
numRowsDelim1=0
numRowsDelim2=0
numRowsDelim3=0

for row in reader:
```

```
    if rownum == 0:
        writer.writerow(("strings_found", str(delim1) +
            "%_error",str(delim2) + "%_error",str(delim3) +
            "%_error"))
    else:
        if int(row[1]) > 0:
            proportion = float(row[2]) / float(row[1]) * 100
            if (proportion > (100 - delim1)) and (proportion < (100
                + delim1)):
                numRowsDelim1 += 1
            if (proportion > (100 - delim2)) and (proportion < (100
                + delim2)):
                numRowsDelim2 += 1
            if (proportion > (100 - delim3)) and (proportion < (100
                + delim3)):
                numRowsDelim3 += 1

    rownum +=1
ifile.close()
writer.writerow((rownum,numRowsDelim1,numRowsDelim2,numRowsDelim3))
exit(0)
```

Appendix I

Figures

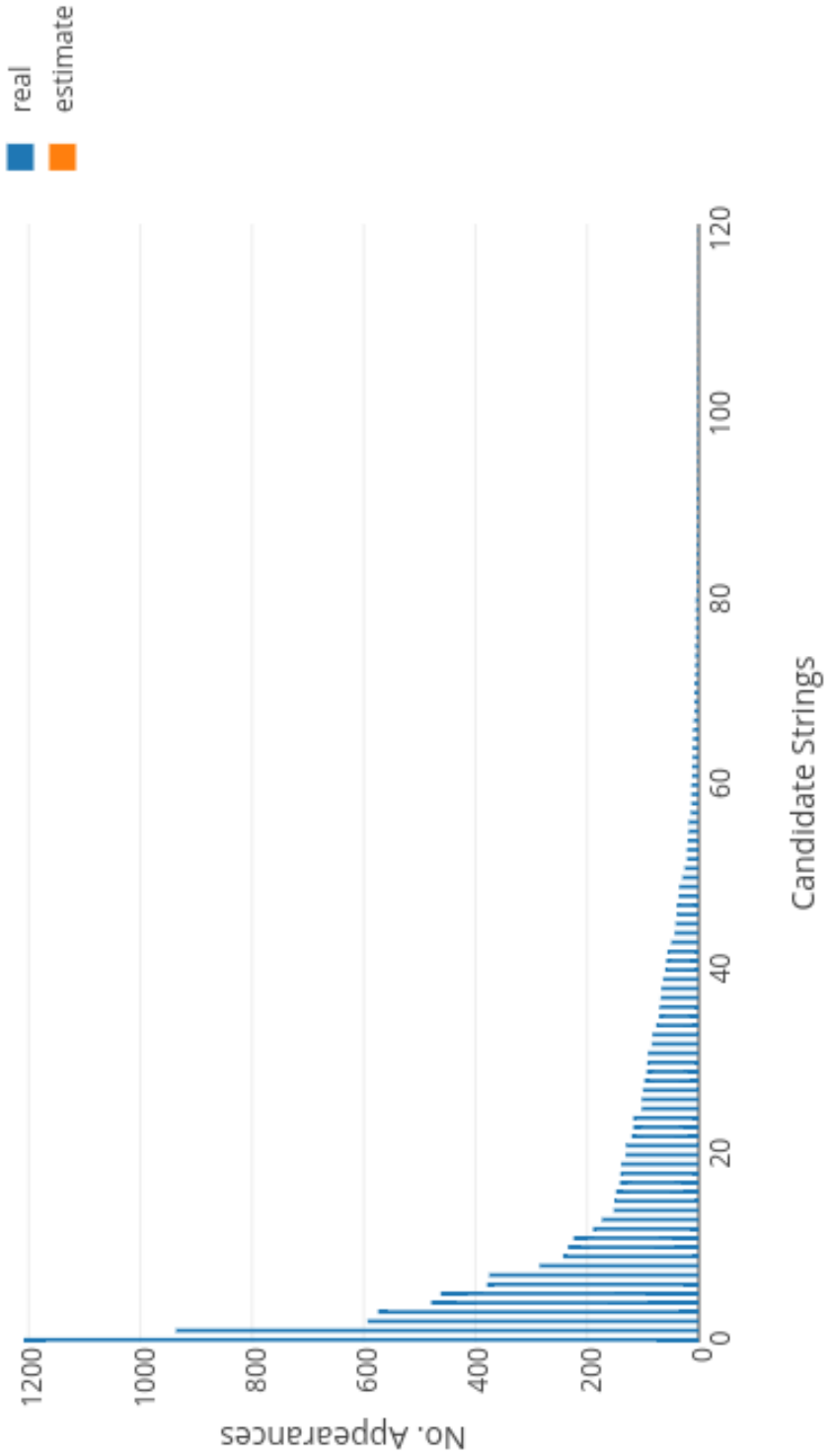

Figure I.1: $\epsilon = 0.1$, population of 10 000 responses

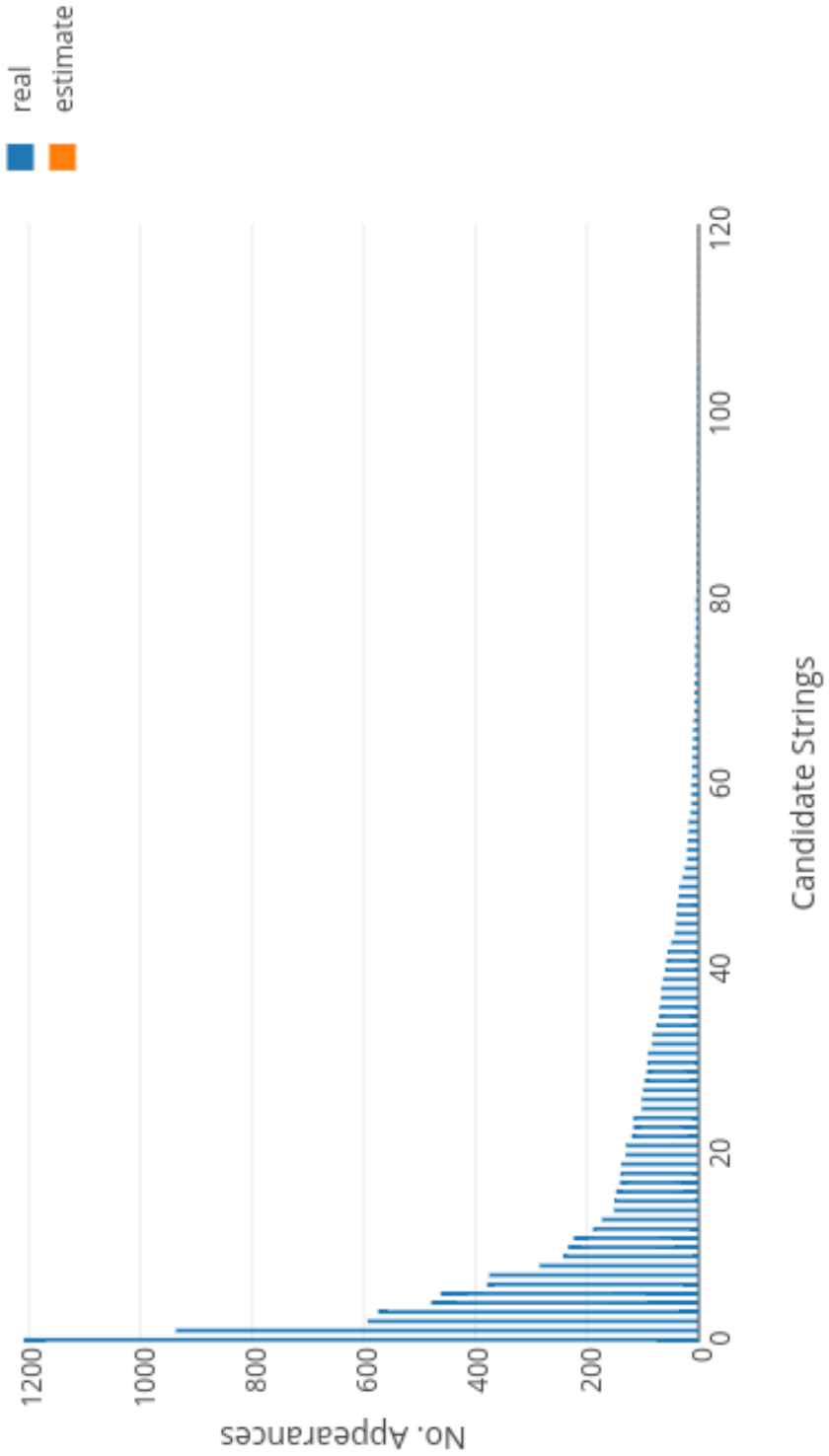

Figure I.2: $\epsilon = 0.1$, population of 100 000 responses

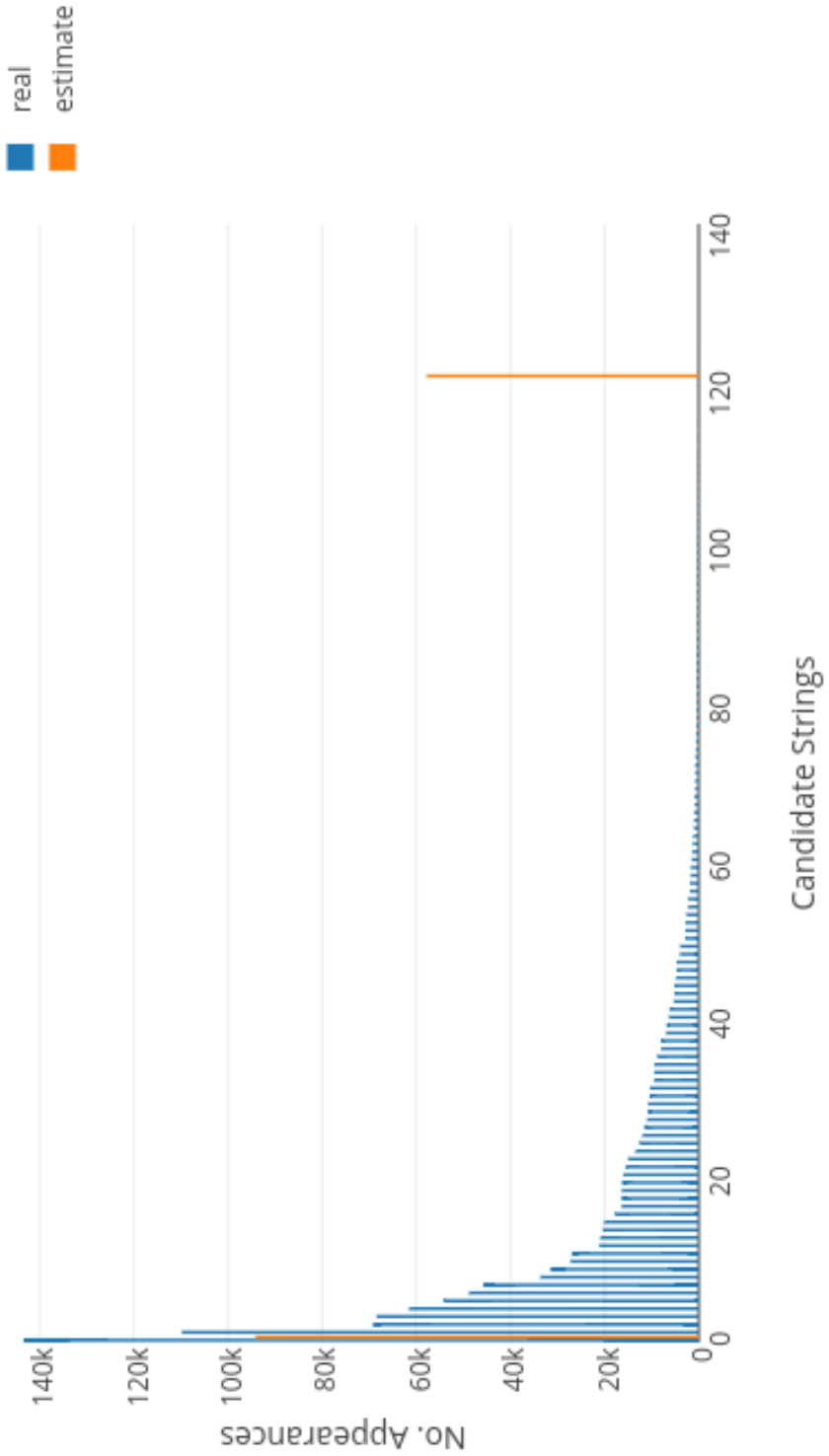

Figure I.3: $\epsilon = 0.1$, population of 1 200 000 responses

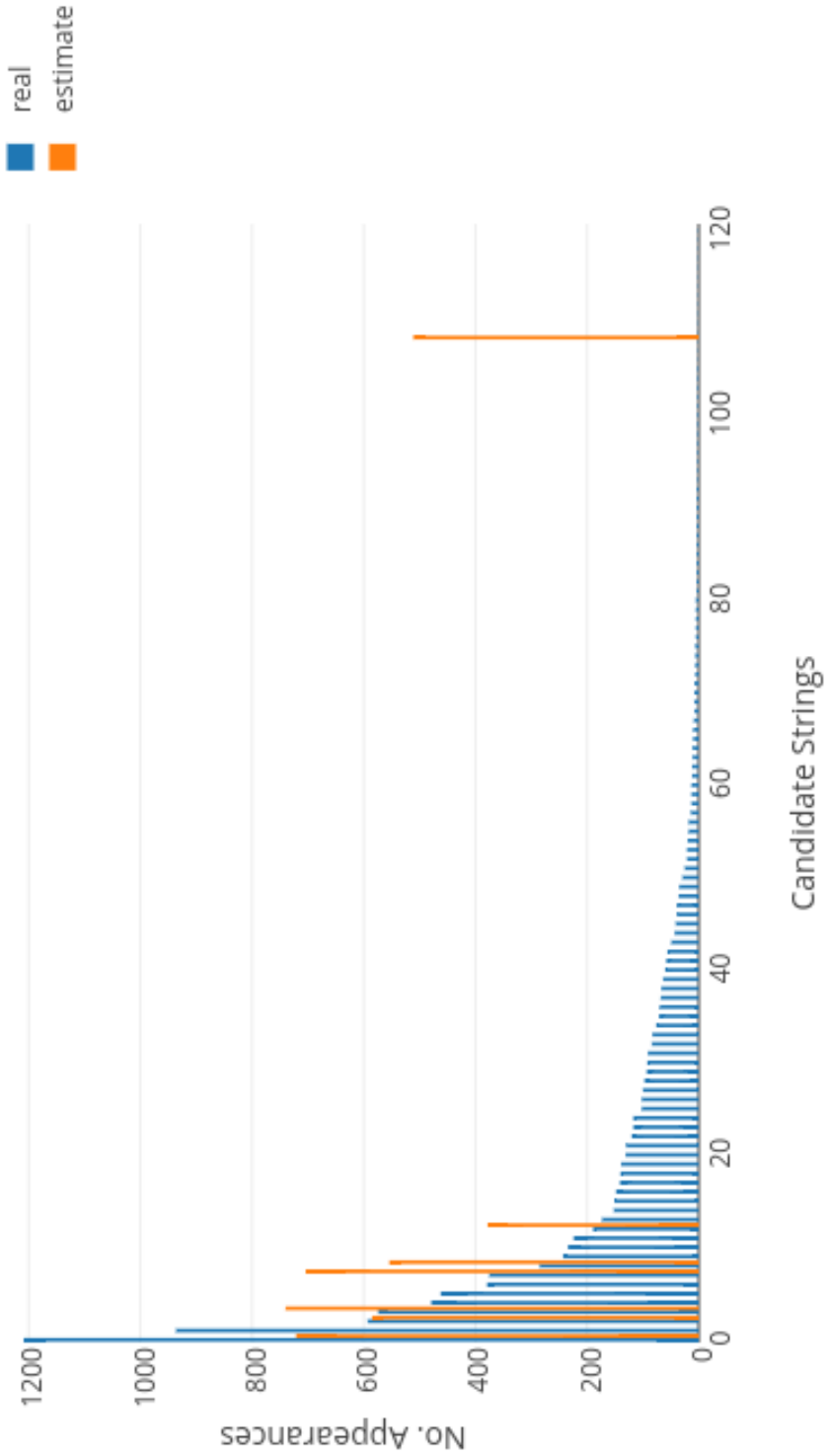

Figure I.4: $\epsilon = 1$, population of 10 000 responses

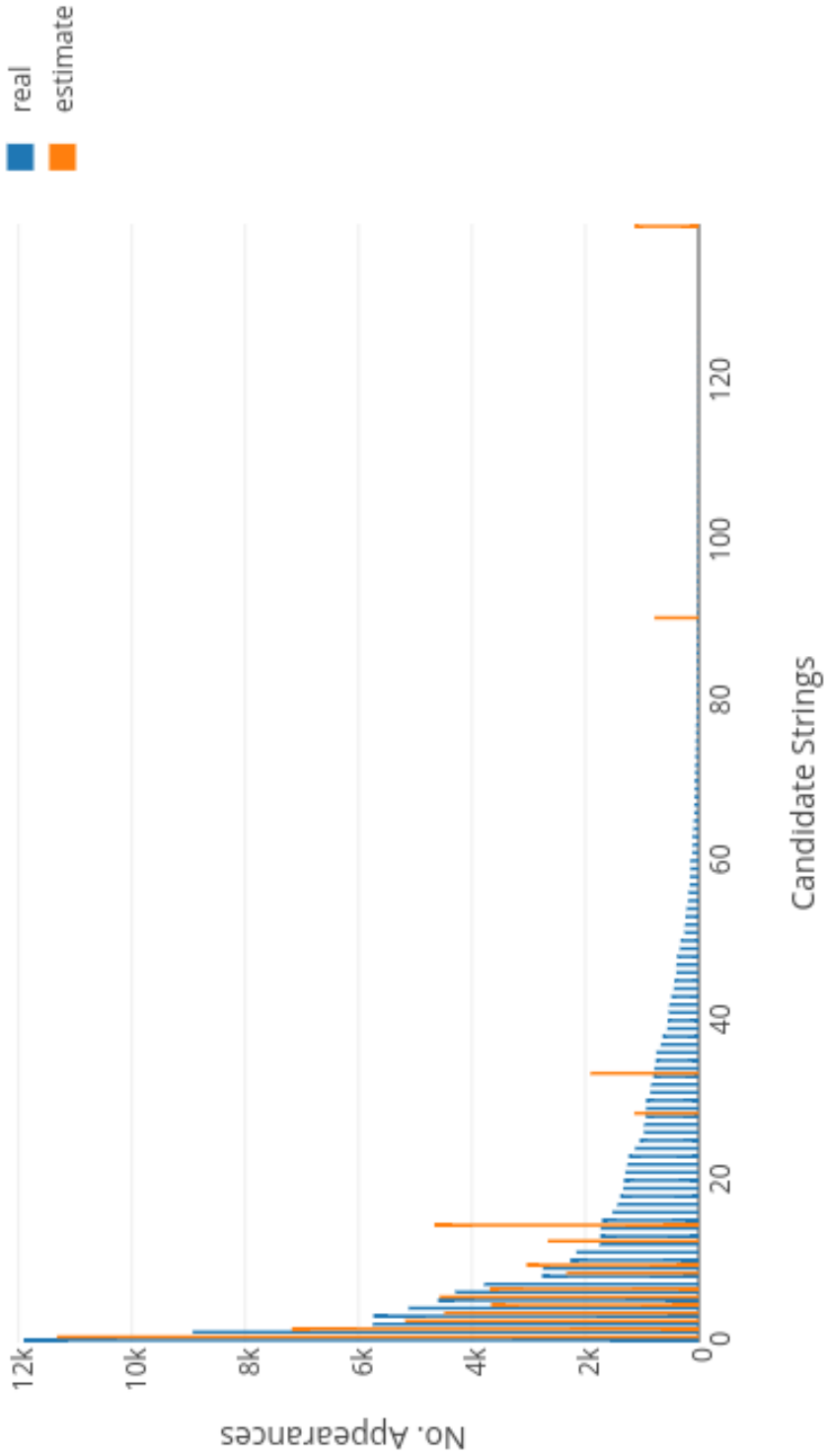

Figure I.5: $\epsilon = 1$, population of 100 000 responses

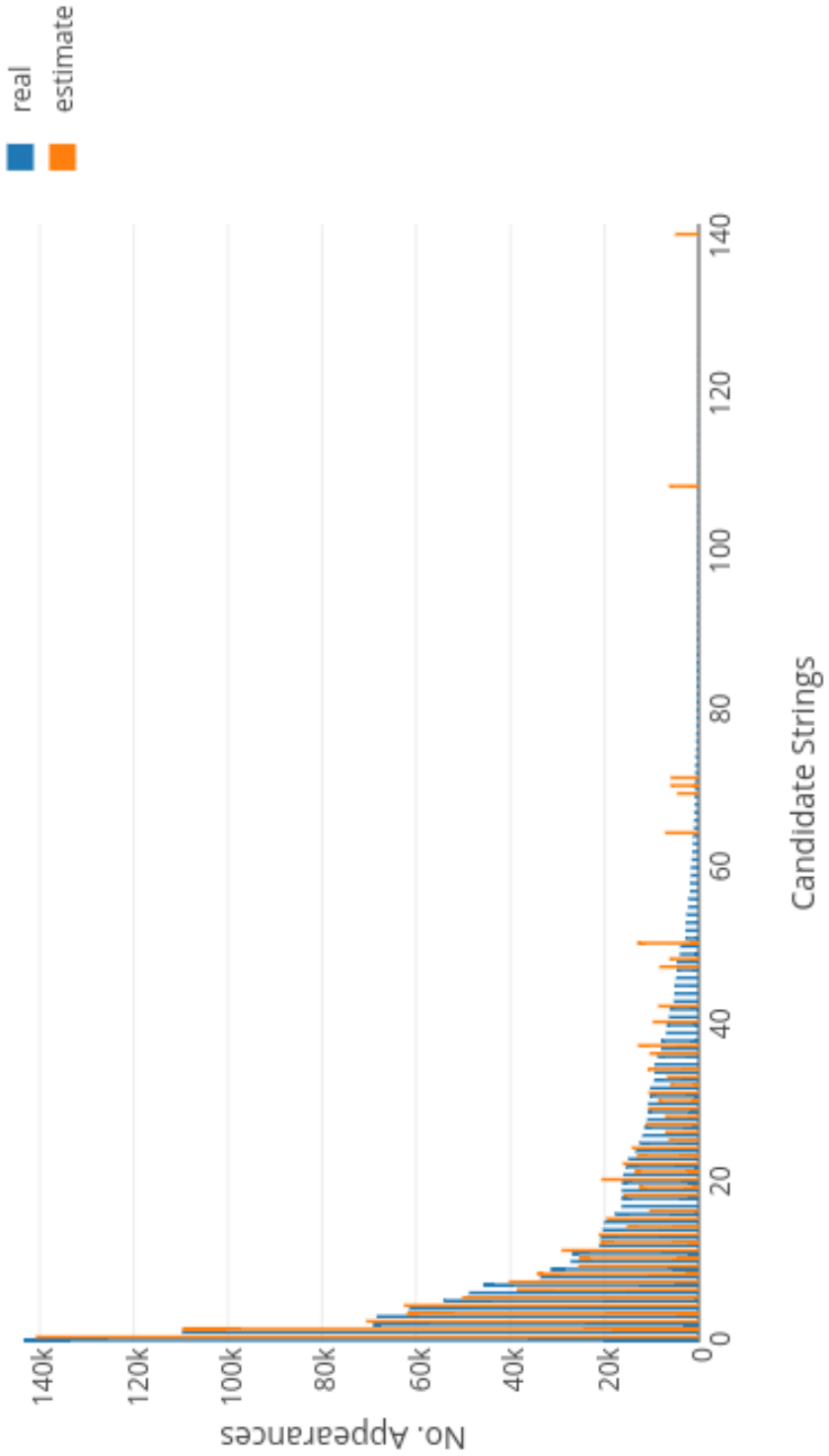

Figure I.6: $\epsilon = 1$, population of 1 200 000 responses

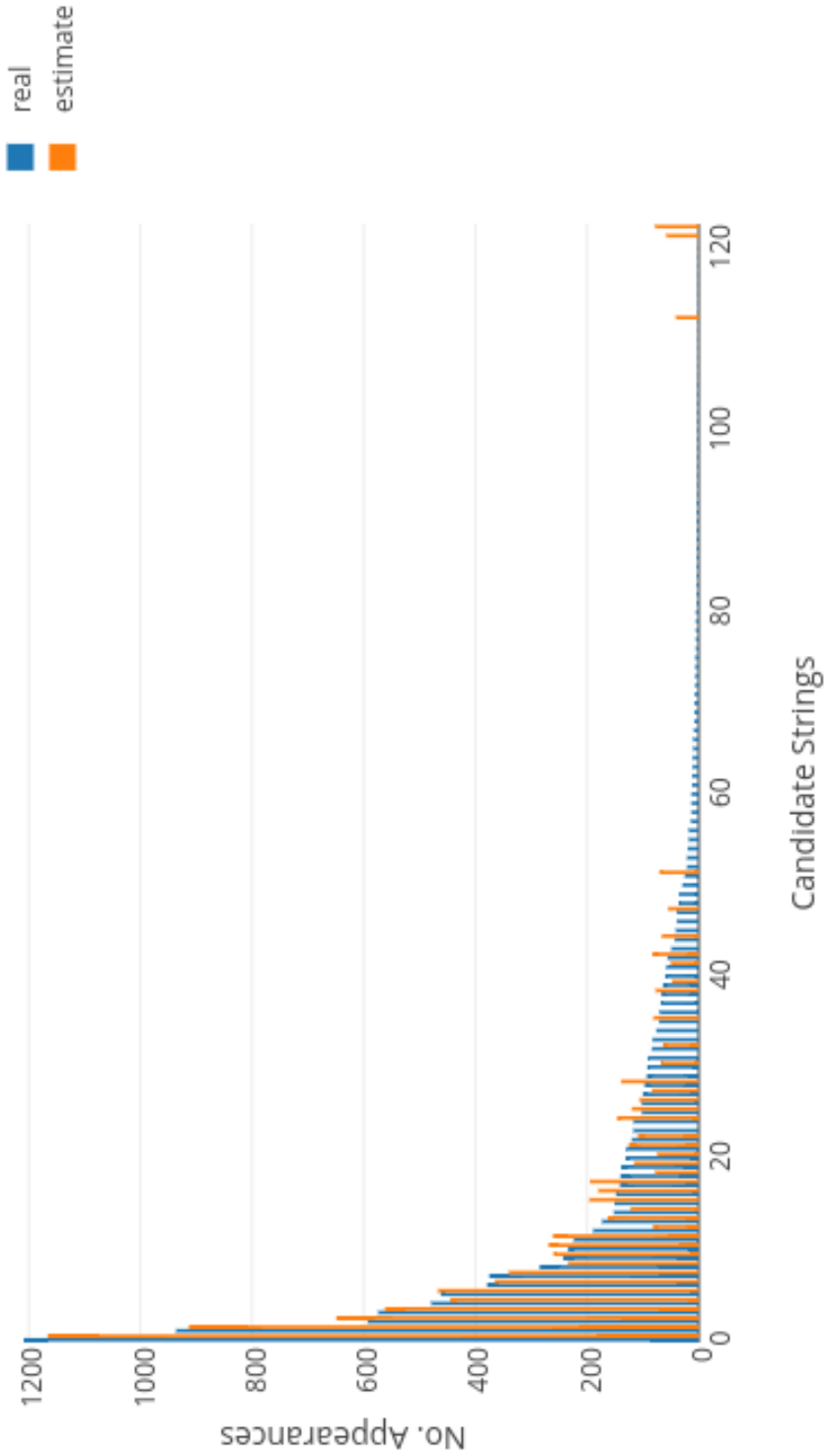

Figure I.7: $\epsilon = 10$, population of 10 000 responses

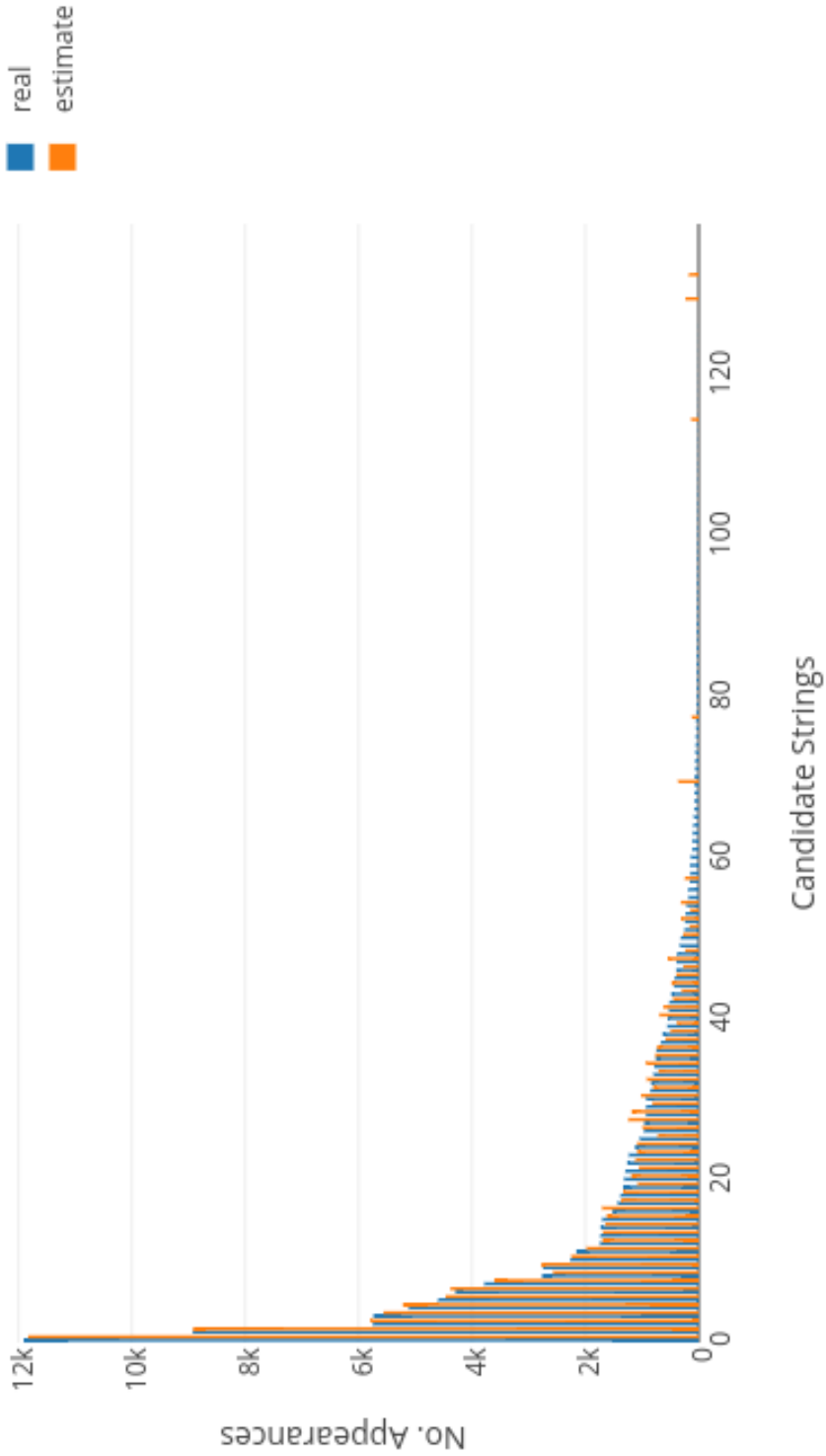

Figure I.8: $\epsilon = 10$, population of 100 000 responses

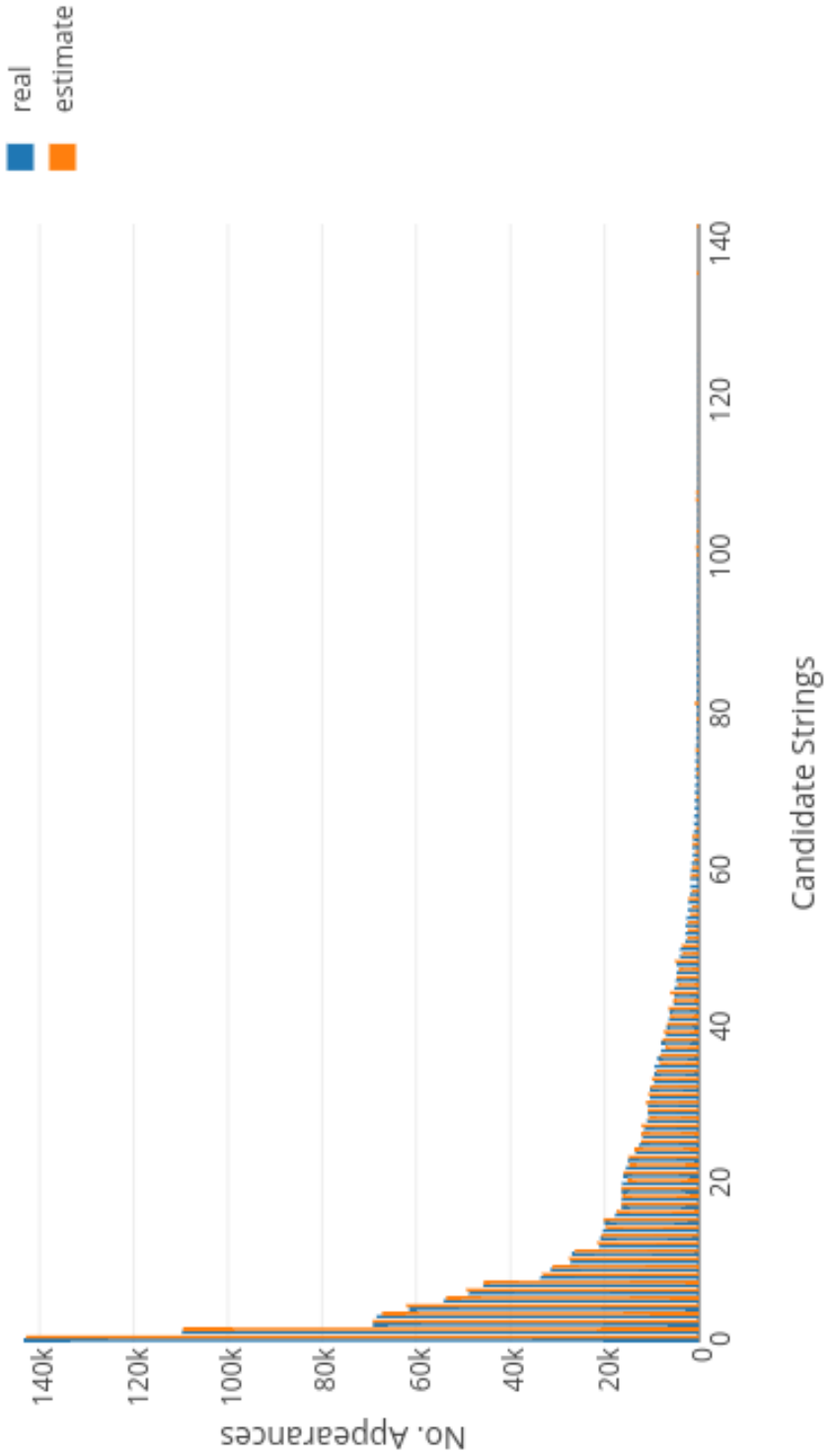

Figure I.9: $\epsilon = 10$, population of 1 200 000 responses