universitätbonn

# OPTIMIZATION OF RANDOMIZED SAMPLING AND RESPONSE FOR DIFFERENTIAL PRIVACY IN COMPLEX QUERIES

MATTHEW LAVENGOOD

Bachelor Thesis

September 11th, 2017

Rheinische Friedrich-Wilhelms-Universität Bonn
Institute of Computer Science 4
Privacy and Security in Ubiquitous Computing Lab

Optimization of Randomized Sampling and Response for Differential Privacy in Complex Queries
Bachelor Thesis
UBIPS-Bsc-0012

Submitted by Matthew Lavengood
Date of submission: September 11th, 2017

First examiner/Erstgutachter: Jun.-Prof. Dr.-Ing. Delphine Reinhardt (née Christin)
Second examiner/Zweitgutachter: Name
Supervisor/Betreuer (Universität Bonn): Jun.-Prof. Dr.-Ing. Delphine Reinhardt (née Christin)
Supervisor/Betreuer (inovex GmbH): Hans-Peter Zorn, Dipl.-Inf.

Rheinische Friedrich-Wilhelms-Universität Bonn
Institute of Computer Science 4
Privacy and Security in Ubiquitous Computing Lab

## ABSTRACT

Differential privacy is a privacy criterion which mathematically guarantees a level of individual privacy while still allowing analysts to obtain aggregate information about a sample. Existing differentially private algorithms often focus on collecting limited and simple statistics such as counts, sums, or simple histograms. In this thesis, I explore the potential of analyzing more complex relationships in a differentially private fashion using randomized sampling and response, an algorithm suitable for use in streaming which satisfies differential privacy as well as zero knowledge privacy. After presenting a procedure for the optimization of randomization parameters with respect to result accuracy, I examine the limits of randomized sampling and response, quantifying the relationship between sample size, proportions of individual histogram buckets, differential privacy level, and result accuracy after parameter optimization. I empirically test these findings by implementing a test system to carry out randomized sampling and response for both single- and multi-column database queries, introducing the possibility of examining relationships and event chains in addition to simple frequency counts. My results represent a refinement in the application of randomized sampling and response and a demonstration of the potential for wider and more flexible utilization of differential privacy.



## ZUSAMMENFASSUNG

Differential Privacy ist ein mathematisches Kriterium, das ein gewisses Schutzniveau für die Privatsphäre einzelner Betroffener garantiert aber dabei die Erkennung statistischer Zusammenhänge zulässt. Viele bisherige Algorithmen, die Differential Privacy erfüllen, setzen darauf, beschränkte und einfache Statistiken wie z.B. Summen, Anzahlen oder einfache Histogramme zu berechnen. In dieser Arbeit untersuche ich Möglichkeiten, komplexere Verhältnisse unter Erfüllung von Differential Privacy zu analysieren, unter Einsatz der Technik "Randomized Sampling and Response", eines für Streaming geeigneten Algorithmus, der sowohl Differential Privacy als auch Zero Knowledge Privacy erfüllt. Neben der Entwicklung einer Prozedur, welche die Randomisierungsparameter bezüglich der resultierenden Genauigkeit der Ergebnisse optimiert, untersuche ich die Grenzen von Randomized Sampling and Response und quantifiziere das Verhältnis zwischen der Stichprobengröße, den Anteilen der individuellen Histogrammgruppen, dem Differential-Privacy-Niveau und der optimierten Genauigkeit der Ergebnisse. Ich überprüfe diese Aussagen empirisch durch die Implementierung eines Testsystems, das Randomized Sampling and Response zur Evaluierung von Datenbankanfragen mit einer oder mehreren Antwortspalten anwendet, und stelle damit die Möglichkeit vor, nicht nur die Häufigkeit einzelner Ereignisse oder Eigenschaften zu erkennen, sondern auch Ereignisketten und Verhältnisse zwischen Variablen zu untersuchen. Meine Ergebnisse stellen eine Verfeinerung sowie eine Erweiterung der Anwendung von Randomized Sampling und Response dar und demonstrieren das Potenzial für einen weitergehenden und flexibleren Einsatz von Differential Privacy.

## ERKLÄRUNG ZUR BACHELOR-THESIS

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

*Bonn, September 11th, 2017*

Matthew Lavengood

# CONTENTS

# 1 INTRODUCTION AND MOTIVATION

In recent years, the volume of data collected digitally about individuals has grown tremendously. This growth has brought with it the potential to analyze personal and machine-generated data on a massive scale. "Big Data" analyses are finding use for diverse purposes ranging from dynamic hotel room pricing [1] to the identification of disease co-occurrence trends [2]. Big Data and Business Analytics is thus a fast-growing industry with revenues projected to exceed $210 billion by 2020 [3].

Although the potential benefits of large-scale personal data analysis are great, they have been saddled with worries relating to the privacy and security of potentially sensitive personal information contained in these data. Public opinion has grown increasingly suspicious of the usage of their online personal data for business purposes, with 80% of social network users indicating concern for the usage of their data by third parties [4].

In addition to its impact on consumer trust and public opinion, the implications of irresponsibly processing personal data are also legal: The protection of individuals' privacy is anchored in EU law under Directive 95/46/EC and national laws (soon to be overhauled when the General Data Protection Regulation enters into force in 2018); in the United States, assorted regulations such as the Health Insurance Portability and Accountability Act (HIPAA) and its privacy provisions must be taken into consideration. Otherwise, individuals and companies analyzing personal data risk punitive fines and sanctions.

In order to temper the legal risks and issues of trust associated with the processing of personal data, Big Data analysts often rely on anonymization, which describes methods which transform data in a way such that it becomes impossible to identify individuals on their basis [5]. However, guaranteeing that a data-set is truly non-identifiable is a non-trivial problem, and many intuitive anonymization techniques have proven fallible upon closer examination (see Section 2.2). As a result, researchers have turned to anonymization methods which rely on rigorous mathematical guarantees rather than the mere perception of anonymity.

One such mathematical guarantee is the concept of differential privacy as introduced by Dwork in 2006. Differential privacy relies on the idea that data are transformed by a randomizing function before being transmitted to the analyst. This randomizing function is considered "differentially private" if, simply put, changing the value of any individual dataset cannot change the value of the output in a predictable way. Since the untrusted analyst only receives the output of the function, the individual's privacy is guaranteed because their individual result cannot noticeably affect the information received by the analyst. However, in aggregate, the output may be used to estimate useful statistics [6].

Differential Privacy is particularly well-suited for streaming applications, where the data to be analyzed changes over time. In contrast to other measures of anonymity such as k-Anonymity and $\ell$-Diversity (see Section 2.3), which focus on the one-time "cleaning" of a static, unchanging database, Differential Privacy is suited to making repeated queries on a database where entries may be changed, added, or removed over time; this is because

a differentially-private algorithm can be applied to an individual database row before it is added to the database.

Although a number of differentially-private algorithms and architectures have been researched, many of these have limitations which constrain their applicability to typical Big Data applications. For instance, some are limited to the computation of specific aggregation statistics, lacking the flexibility to be used for the collection of more complex information, such as the sequence of websites visited during a session. Others require the presence of a "trusted curator" or "trusted nodes" which have access to the original, personalized data in its entirety; however, in the case of data analysts who wish to analyze user browsing histories or clickstreams, such a trusted actor is not likely to exist.

In this thesis, I examine the possibility of expanding the application of the existing differentially-private algorithm "randomized sampling and response", proposed and implemented by Do Le Quoc in 2016 within the framework of his PrivApprox system. Randomized sampling and response allows for differentially-private, streaming collection of data in a distributed fashion. A distributed algorithm is advantageous since it relies on the original personal data only being collected on the individual users' devices instead of on a central database. These devices then respond to queries from an analyst with responses which have already been anonymized; this allows the central aggregator to calculate the desired statistics, while ensuring that the identifiable personal data never leave the client device, thus rendering the "trusted curator" unnecessary. In the algorithm itself (randomized sampling and response), each bit in the subject's response to a query has a pre-determined chance of being a random answer rather than an honest one, and each subject flips a coin to decide whether to answer at all. In addition to being differentially-private, this method also satisfies zero knowledge privacy, a more strict privacy notion introduced by Gehrke [7].

So far, the published technical report on PrivApprox only tests the collection of simple aggregation statistics [8]; in this thesis, I examine how more complex relationships can be analyzed. Using the Java programming language, I implement a test system which collects sample data from the Twitter API in conjunction with the wrapper twitter4j and enables the query of statistics using SQLite, applying the randomized sampling and response method. I evaluate its ability to flexibly execute multi-column SQLite queries in a differentially-private manner. In addition, I theoretically and empirically analyze the amount of error introduced through the randomized response mechanism, and introduce a procedure to optimize the algorithm's parameters, minimizing the theoretical variance for a desired privacy level.

I begin with a more detailed theoretical introduction to the concepts of anonymization and mathematical notions of anonymity, including differential privacy and zero knowledge privacy. I also introduce the randomized response algorithm and its mathematical properties. After a review of the existing literature of differentially-private algorithms, I introduce my technical implementation for the collection, randomization, and analysis of test data, including new procedures for the optimization of randomization parameters and the processing of queries including multiple responses. I then test the performance of my implementation on a set of devised test cases and evaluate these on the basis of their variance, as related to the level of differential privacy (and zero knowledge privacy) provided and the theoretically expected variance.

# 2 BACKGROUND

## 2.1 ANONYMIZATION

### 2.1.1 *Definition*

Anonymization is the principle method allowing the harvesting of useful statistics from personal data while eliminating the risk to individual privacy as well as the need to comply with legal regulations on the handling of personal data. Specificylly, Recital 26 to the General Data Protection Regulation excepts anonymous data as well as "personal data rendered anonymous" (i.e. anonymized data) from its stipulations and defines them as follows:

**Definition 2.1.1** (Anonymized data)**.** Personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable. [5]

Similar definitions of anonymous data may be found in §164.502(d) of HIPAA under the moniker "de-identification" [9].

In the following sections, well-known methods of anonymization are described and evaluated.

## 2.2 NAIVE ANONYMIZATION AND ITS WEAKNESSES

The most intuitive and simple solution to anonymize personal data is to remove obvious identifying characteristics such as names, telephone numbers, and addresses. This is the anonymization method explicitly recommended by §164.514 of HIPAA [9]. However, such anonymization has proven to be surprisingly unreliable. It is often possible to determine the identity of individual "anonymous" entries by comparing common attributes in other public databases. Such a "linking attack" aims to find a set of attributes which reveal enough information to narrow the identity of an entry to one specific person [10].

Several case studies have been published to demonstrate the inadequacy of simply removing identifiers. In 1997, Sweeney successfully re-identified anonymized health records of Governor William Weld of Massachusetts using only his gender, date of birth, and ZIP code by matching this information to public voting records. Sweeney also estimated that this combination of information is enough to identify 87 percent of the US population [11]. In 2013, Sweeney published a further case study showing that 35 of 81 cases in news articles were able to be definitively linked to health records in public state hospital data, using a combination of gender, age, general address, hospital, and incident details [11].

Furthermore, in 2006, Ohm succeeded in identifying the users associated with ostensibly anonymized publicized Netflix ratings via linkage with public ratings in the Internet Movie Database. Although such a privacy breach may seem relatively harmless at first glance, the authors point out that the ratings of political, religious, or LGBT-themed media can be used to glean highly sensitive information about the subject [12]. Indeed,

this incident led one individual to bring a lawsuit against Netflix in 2009 who claimed that the release of her viewing history allowed the identification of her sexual orientation [13].

More recently, as reported in August 2017, journalist Svea Eckert and data scientist Andreas Dewes were able to obtain a database of clickstreams from a data broker consisting only of URLs and timestamps from German internet users. They thus obtained sensitive information such as the adult media viewed by a judge, the medication used by a German MP, and even details on a German cybercrime investigation [14].

These cases make clear that simply removing immediate identifiers is not sufficient to ensure the privacy of data subjects.

## 2.3 MATHEMATICAL CRITERIA FOR ANONYMIZED DATASETS

The weaknesses of basic anonymization have led researchers to devise more rigorous mathematical guarantees of privacy. These include a number of measures such as k-Anonymity, $\ell$-Diversity, t-Closeness, $\delta$-Presence and $\epsilon$-Differential Privacy [15].

It should be noted that it is impossible to completely prevent information about an individual from being learned if a statistical analysis is to be useful. For instance, a study demonstrating that smoking causes cancer will inevitably reveal that any individual smoker is more likely to become ill with cancer. As a result, any criteria for privacy must be defined by some numerical constraint which sets "how much" we are allowed to learn from a given database entry [6]. The following sections introduce some of the devised mathematical criteria for anonymity.

### k-ANONYMITY

k-Anonymity was introduced by Samarati and Sweeney in 1998 as a measure against the linkage attacks described in the previous section. It is defined as follows [16]:

**Definition 2.3.1** (k-Anonymity)**.** A dataset fulfills k-Anonymity if every combination of values of quasi-identifiers can be indistinctly matched to at least k individuals.

This ensures that no single entry can be identified as belonging to a particular individual, since any combination of potentially-identifying features are shared by at least $k-1$ other entries. This set of k or more entries is said to form an "equivalence class." This renders impossible the identification of a particular row belonging to an individual.

Although k-Anonymity eliminates the possibility of de-anonymization of individual database records, it does not eliminate all risk of deducing compromising information from a database. Of particular interest is the *homogeneity attack*: if an attacker knows that her target, a 45-year-old male from the ZIP code group 123**, is contained in a public medical database, and finds that all k males in the database sharing these quasi-identifiers were diagnosed with lung cancer, then she can deduce that the target also has lung cancer. The same risk can be posed if the entries all have similar diagnoses, an unusual distribution of diagnoses, or if the attacker possesses additional background knowledge which allows her to identify which diagnoses could plausibly apply to the target [17]. $\ell$-Diversity and t-Closeness are two criteria devised to address these concerns.

### $\ell$-DIVERSITY

$\ell$-Diversity aims to prevent such information deductions by ensuring a degree of variation within each equivalence class. It can be defined as follows[17]:

**Definition 2.3.2** (ℓ-Diversity)**.** A set of entries sharing the same quasi-identifiers (equivalence class) is ℓ-diverse if it contains at least ℓ "well-represented"[1] values for the sensitive attribute S. A table is ℓ-diverse if every equivalence class in the table is ℓ-diverse.

By ensuring that there is no equivalence class where all members share the same sensitive piece of information, ℓ-Diversity protects against a homogeneity attack. It also provides security against a background knowledge attack, dependent on the value of ℓ: in order to derive the sensitive information of one individual in the equivalence class, an attacker must eliminate at least $\ell - 1$ other possibilities, a task which becomes increasingly difficult with an increasing ℓ [17].

ℓ-Diversity remains vulnerable to *skewedness attacks*: even if an equivalence class is ℓ-diverse, its distribution of values may still be significantly deviant from the norm such that sensitive information can still be determined. For instance, if a database shows ℓ distinct salary ranges under 30,000, then its ℓ-Diversity will not prevent an attacker from learning that any member of that class has a relatively low salary [18].

t-CLOSENESS t-Closeness is a measure to prevent against a skewedness attack. It stipulates that the value distribution in any equivalence class must be relatively similar to that of the entire table. Formally [18]:

**Definition 2.3.3** (t-Closeness)**.** An equivalence class is said to have t-Closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-Closeness if all equivalence classes have t-Closeness.

With t-Closeness and an appropriate choice of t, a database is adequately protected against a skewedness attack. Since any grouping of equivalent entries has a similar distribution of results, one cannot draw any specific conclusions about any member of an equivalence group using the database.

It must be noted that the above-described criteria of k-Anonymity, ℓ-Diversity, and t-Closeness (as well as further criteria such as δ-Presence [19]) focus on the "cleaning" of a static, intact database with full personal details for anonymized release. This introduces the problem of a "trusted curator," which is not a part of my problem setting.

In addition, these criteria face difficulty being implemented in a streaming setting where each incoming data point should be anonymized immediately upon storage. This is because the properties can only be satisfied with regard to the entire database; a single arriving data point cannot be rendered k-anonymous or ℓ-diverse in isolation. Although an example for a proposed streaming algorithm for k-anonymity can be found [20], the additional implementation of ℓ-Diversity and t-Closeness on a streaming basis could not be found in the literature.

1 The exact meaning of "well-represented" is flexible and must be determined by the implementor. In essence, each value contributing to ℓ-diversity must be frequent enough in the equivalence class that it would not be negligibly improbable from the perspective of an attacker. If, say, 10 distinct sensitive values are present in the equivalence class, but 9 of the values occupy only 1% of the entries, then it should not be described as 10-diverse, since the infrequent values offer no real protection against a homogeneity attack [17].

## 2.4 DIFFERENTIAL PRIVACY

### 2.4.1 *Definition*

$\epsilon$-Differential Privacy is a privacy criterion proposed by Dwork in 2010 [6] which has proven itself to be more applicable to streaming and distributed situations.

Intuitively, differential privacy is a requirement that the output of a data analysis mechanism be relatively independent of the content of any individual data row. Formally, differential privacy is defined by Dwork as follows [6]:

**Definition 2.4.1** (Differential Privacy)**.** A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is $\epsilon$-differentially private if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\|_1 \leqslant 1$:

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leqslant \exp(\epsilon)\Pr[\mathcal{M}(y) \in \mathcal{S}]$$

In other words, a $\epsilon$-differentially private algorithm over a data table will not become much more likely to produce any given result if we change one data item in it. The degree to which an individual data row is allowed to influence the randomized distribution of the result is determined by the variable $\epsilon$.

In its simplest conception, differential privacy consists of a trusted curator who maintains a database with private information and a differentially-private mechanism which allows untrusted analysts to perform statistical evaluations without being able to discern any individual values. However, as previously discussed, this model is insufficient in many circumstances.

To address this problem, many newer differential privacy solutions rely on the application of the differential privacy mechanism at the client's device such that the data collector never learns the "real" data at all [8, 21, 22]. Erlingsson's differentially-private RAPPOR algorithm has been implemented for use in Google Chrome [21]. Apple also announced the implementation of differential privacy in iOS 10 [23]. Although few details on the specific algorithm can be found publicly, it seems similar to the RAPPOR algorithm used by Google Chrome [24].

## 2.5 RANDOMIZED RESPONSE

#### 2.5.0.1 *Description*

One early differentially-private algorithm, which finds application in this thesis, is *randomized response*. Randomized response was introduced by Warner in 1965 [25] in response to the problem that many survey respondents are unwilling to give honest answers to embarrassing survey questions due to a lack of trust for the interviewer or simple natural reluctance.

To mitigate this issue, Warner proposed a system in which respondents are given two contradictory versions of a potentially embarrassing statement, for instance "I have driven under the influence of alcohol" and "I have never driven under the influence of alcohol." Respondents are then asked to flip a coin (or use other randomized means) to determine which query to respond to. The surveyor does not learn the result of the coinflip, and thus does not know which question was answered. With knowledge of

the probability distribution of the randomization [1] (and under the assumption that all respondents followed the procedure faithfully), it is then possible for the researcher to estimate the true proportions of individuals who have driven under the influence, but it cannot be proven that any particular individual has admitted to it; each respondent has plausible deniability since it is not known *which* query was answered [25].

An alternative model described by Dwork [6] and utilized in PrivApprox (see Section 3) [8] relies on a two-step procedure which can be defined as follows:

**Definition 2.5.1** (Randomized Response)**.** The randomized response algorithm $\mathcal{M}$ takes as a truthful answer as an input, and outputs an answer according to the following procedure:

1. Flip a coin. If heads, respond truthfully. If tails, go to Step 2.
2. Flip a coin. If heads, respond "yes." If tails, respond "no."

$p$ and $q$ define the probabilities of receiving heads in the first and second step respectively.

#### 2.5.0.2 *Differential Privacy Characteristics of Randomized Sampling and Response*

Randomized response as defined above is differentially private. Using Definition 2.6.1, let us observe the answer of a single respondent. Let the domain and range be described as $\{0, 1\}$, where 0 indicates "False" and 1 indicates "True"; the domain indicates the truth, while the range indicates the result of following the randomized response procedure.

Since our domain and range only consists of two entries each, we only need to prove the following:

$$\Pr[\mathcal{M}(1) = 1] \leqslant \exp(\epsilon)\Pr[\mathcal{M}(0) = 1] \tag{2.1}$$
$$\Pr[\mathcal{M}(0) = 1] \leqslant \exp(\epsilon)\Pr[\mathcal{M}(1) = 1] \tag{2.2}$$
$$\Pr[\mathcal{M}(1) = 0] \leqslant \exp(\epsilon)\Pr[\mathcal{M}(0) = 0] \tag{2.3}$$
$$\Pr[\mathcal{M}(0) = 0] \leqslant \exp(\epsilon)\Pr[\mathcal{M}(1) = 0] \tag{2.4}$$

We can calculate the respective probabilities as follows:

$$\begin{aligned}
\Pr[\mathcal{M}(1) = 1] = \Pr[Answer = \text{"Yes"} | Truth = \text{"Yes"}] &= p + (1-p)q \\
\Pr[\mathcal{M}(0) = 1] = \Pr[Answer = \text{"Yes"} | Truth = \text{"No"}] &= (1-p)q \\
\Pr[\mathcal{M}(1) = 0] = \Pr[Answer = \text{"No"} | Truth = \text{"Yes"}] &= (1-p)(1-q) \\
\Pr[\mathcal{M}(0) = 0] = \Pr[Answer = \text{"No"} | Truth = \text{"No"}] &= p + (1-p)(1-q)
\end{aligned}$$

From these values, it is apparent (and intuitive) that $\Pr[\mathcal{M}(0) = 1] \leqslant \Pr[\mathcal{M}(1) = 1]$ and $\Pr[\mathcal{M}(1) = 0] \leqslant \Pr[\mathcal{M}(0) = 0]$. Thus, equations (2.2) and (2.3) above are fulfilled for $\epsilon = 0$.

For equation (2.1):

$$\epsilon \geqslant \ln\left(\frac{\Pr[\mathcal{M}(1) = 1]}{\Pr[\mathcal{M}(0) = 1]}\right) \geqslant \ln\left(\frac{p + (1-p)q}{(1-p)q}\right)$$

For equation (2.4):

$$\epsilon \geqslant \ln\left(\frac{\Pr[\mathcal{M}(0) = 0]}{\Pr[\mathcal{M}(1) = 0]}\right) \geqslant \ln\left(\frac{p + (1-p)(1-q)}{(1-p)(1-q)}\right)$$

1 For mathematical reasons, the randomization in Warner's original method may not be a "fair" coin flip with a 50% probability of each result.

Thus, the two-step randomized response algorithm is $\epsilon$-differentially private, where [8] :

$$\epsilon_{RR} = \max\left(\ln\left(\frac{p+(1-p)q}{(1-p)q}\right), \ln\left(\frac{p+(1-p)(1-q)}{(1-p)(1-q)}\right)\right)$$
$$\epsilon_{RR} = \ln\left(\max\left(\frac{p+(1-p)q}{(1-p)q}, \frac{p+(1-p)(1-q)}{(1-p)(1-q)}\right)\right) \tag{2.5}$$

This can be further generalized to an algorithm $\mathcal{M}'$ which applies the algorithm $\mathcal{M}$ to each element in a set of "yes"/"no" responses X, outputting a set of randomized "yes"/"no" responses B. Because each randomization is independent of the others, this composition is also differentially private, since:

$$\begin{aligned}
&\Pr[\mathcal{M}(x_i) = b_i] \leqslant \exp(\epsilon)\Pr[\mathcal{M}(x_i) \neq b_i] \\
\Leftrightarrow &\Pr[\mathcal{M}(x_i) = b_i] \cdot \prod_{\substack{j=1 \\ j\neq i}}^{n} \Pr[\mathcal{M}(x_j) = b_j] \leqslant \exp(\epsilon)\Pr[\mathcal{M}(x_i) \neq b_i] \cdot \prod_{\substack{j=1 \\ j\neq i}}^{n} \Pr[\mathcal{M}(x_j) = b_j] \\
\Leftrightarrow &\Pr[\mathcal{M}'(X) = B] \leqslant \exp(\epsilon)\Pr[\mathcal{M}'(X) = B']
\end{aligned}$$

where $B'$ indicates an output which differs in only one element from B.

The differential privacy of the algorithm can also be augmented using random-sampling: Any given $\epsilon$-differentially private algorithm becomes $\epsilon'$-differentially private when sampling with a sampling rate $s$ is introduced, where [26]:

$$\epsilon' = \ln\left(1 + \left(s\left(\exp\left(\epsilon\right) - 1\right)\right)\right)$$

It follows that randomized response combined with random sampling yields a differential privacy factor of:

$$\epsilon'_{RR} = \ln\left(1 + \left(s\left(\max\left(\frac{p+(1-p)q}{(1-p)q}, \frac{p+(1-p)(1-q)}{(1-p)(1-q)}\right) - 1\right)\right)\right) \tag{2.6}$$

#### 2.5.0.3 *Result Estimation*

After receiving a set of randomized responses, the analyst must be able to estimate the true underlying statistic $r$, the proportion of respondents with honest "true" answers.

Let $Y_r$ indicate the number of positive responses in the output. Additionally, let $N'$ be the total number of responses after the application of randomized sampling. The expected number of positive responses will consist of approximately $p \cdot r \cdot N'$ honest "yes" responses as well as approximately $(1-p) \cdot q \cdot N'$ randomized "yes" responses. The resulting equation can then be solved for $r$ in order to receive an estimate $\hat{r}$ for the true proportion:

$$\begin{aligned}
&Y \approx p \cdot r \cdot N' + (1-p) \cdot q \cdot N' \\
\Leftrightarrow &\hat{r} = \frac{Y - (1-p) \cdot q \cdot N'}{p \cdot N'}
\end{aligned} \tag{2.7}$$

#### 2.5.0.4 *Accuracy of Result Estimation*

In the following, in order to characterize the accuracy (also called "utility") of the estimated result of randomized response, I estimate the standard deviation of the estimate of $r$.

$p$, $r$, and $q$ are constants. $N'$ is a randomized variable which can be characterized by the binomial distribution, since each respondent constitutes one trial with the probability $s$ of responding. If $N$ is the total number of potential respondents, then $N'$ has a variance of $Ns(1-s)$ and a standard deviation of $\sqrt{Ns(1-s)}$. This can be divided by the expected value of $N'$, $sN$, to obtain the relative standard deviation, $\frac{\sqrt{Ns(1-s)}}{Ns} = \sqrt{\frac{1-s}{Ns}}$. Since the relative variance becomes very small for a large $N$ and a normal (not too small) value of $s$, I will regard $N'$ as a constant for the following analysis.

Assuming $N'$ is constant, $Y$ may be regarded as the sole randomized variable determining $r$. $Y$ follows the binomial distribution, since one can consider each sample as a randomized trial with the probability $p \cdot r + (1-p) \cdot q$ of delivering the response "Yes." Thus, the variance of $Y$ is $N'(p \cdot r + (1-p) \cdot q)(1 - (p \cdot r + (1-p) \cdot q))$.

The variance of equation (2.7) can thus be calculated as [27]:

$$\begin{aligned} V(\hat{r}) &= \frac{N'(p \cdot r + (1-p) \cdot q)(1 - (p \cdot r + (1-p) \cdot q))}{p^2 \cdot N'^2} \\ V(\hat{r}) &= \frac{(p \cdot r + (1-p) \cdot q)(1 - (p \cdot r + (1-p) \cdot q))}{p^2 \cdot N'} \\ V(\hat{r}) &\approx \frac{(p \cdot r + (1-p) \cdot q)(1 - (p \cdot r + (1-p) \cdot q))}{p^2 \cdot s \cdot N} \end{aligned} \tag{2.8}$$

Although the absolute standard deviation $\sqrt{V(\hat{r})}$ increases as the accurate proportion $r$ increases, the coefficient of variation $\frac{\sqrt{V(\hat{r})}}{r}$ decreases. Since utility is generally evaluated as $\frac{accurate-estimated}{accurate}$, the coefficient of variation, i.e. the ratio of standard deviation to the accurate value, is more appropriate as a measure of utility and will be utilized as such during the evaluation portion.

### 2.5.1 *Vulnerability in Repeated Queries*

One weakness of differential privacy is that it degrades if the same query is submitted multiple times. Intuitively, this is clear: If, say, a subject answers the same question many times in a row using randomized response, then the true answer will become increasingly predictable with each answer (by taking the average of all estimates).

Mathematically, the execution of two differentially-private mechanisms on the same database can be expressed as the composition of the two mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ into a

single mechanism $\mathcal{M}_{12}(x) = (\mathcal{M}_1(x), (\mathcal{M}_2(x))$. Then, for inputs $x$ and $y$ with $\|x - y\|_1 \leqslant 1$ we have [6]:

$$\begin{aligned}
& \Pr[\mathcal{M}_{12}(x) = (r_1, r_2)] \leqslant && \exp(\epsilon_{12}) \cdot \Pr[\mathcal{M}_{12}(y) = (r_1, r_2)] \\
\Leftrightarrow\ & \Pr[\mathcal{M}_1(x) = r_1] \cdot \Pr[\mathcal{M}_2(x) = r_2] \leqslant && \exp(\epsilon_{12}) \cdot \Pr[\mathcal{M}_1(y) = r_1] \cdot \Pr[\mathcal{M}_2(y) = r_2] \\
\Leftrightarrow\ & \exp(\epsilon_{12}) \geqslant && \frac{\Pr[\mathcal{M}_1(x) = r_1] \cdot \Pr[\mathcal{M}_2(x) = r_2]}{\Pr[\mathcal{M}_1(y) = r_1] \cdot \Pr[\mathcal{M}_2(y) = r_2]} \\
\Leftrightarrow\ & \exp(\epsilon_{12}) \geqslant && \exp(\epsilon_1) \cdot \exp(\epsilon_2) \\
\Leftrightarrow\ & \exp(\epsilon_{12}) \geqslant && \exp(\epsilon_1 + \epsilon_2) \\
\Leftrightarrow\ & \epsilon_{12} \geqslant && \epsilon_1 + \epsilon_2
\end{aligned} \tag{2.9}$$

This can be extended to conclude that $\epsilon$ composes additively for multiple differentially-private queries with the same input. As a result, repeated queries over the same data must be avoided for the differential privacy factor to remain acceptable.

## 2.6 ZERO-KNOWLEDGE PRIVACY

Zero-knowledge privacy is a notion of privacy introduced by Gehrke [7] in 2011. Zero-knowledge privacy is meant to offer an additional layer of privacy where differential privacy may not be effective, particularly in cases where information about other persons *associated* with an individual may already compromise that individual's privacy. This is of particular interest in social networks; aggregate characteristics about the friends of an individual can be used to detect that individual's characteristics with high accuracy. For instance, if a differentially-private mechanism is used to determine that a clique in an American social network mostly consists of Republicans, then it can be guessed that any individual in that clique is a Republican with high probability, even if the individual did not volunteer this information [7].

Zero-knowledge privacy offers a solution to this issue by allowing the information released by a mechanism to be compared to a specific set of information considered "acceptable". Formally, let $\mathrm{Out}_A(A(z) \leftrightarrow \mathcal{M}(D))$ denote the output received by an analyst $A$ after interacting with a randomized algorithm $\mathcal{M}$ over the database $D$. Furthermore, define $agg$ as a class of randomized algorithms providing aggregate information. Within this framework, zero-knowledge privacy is defined as follows:

**Definition 2.6.1** (Zero Knowledge Privacy)**.** $\mathcal{M}$ is $\epsilon$-zero-knowledge private with respect to $agg$ if there exists a $T \in agg$ such that for every adversary $A$, there exists a simulator $S$ such that for every database $D$, every $z \in \{0,1\}^*$ , every integer $i \in [n]$, and every $W \subseteq \{0,1\}^*$ , the following hold:

$$\begin{aligned}
\Pr[\mathrm{Out}_A(A(z) \leftrightarrow \mathcal{M}(D)) \in W] &\leqslant \exp(\epsilon) \cdot \Pr[S(z, T(D_{-i}), i, n) \in W] \\
\exp(\epsilon) \cdot \Pr[S(z, T(D_{-i}), i, n) \in W] &\leqslant \Pr[\mathrm{Out}_A(A(z) \leftrightarrow \mathcal{M}(D)) \in W]
\end{aligned}$$

where $D_{-i}$ represents the database $D$ with $i$-th dataset concealed.

Instead of comparing the result of $\mathcal{M}$ to the result of $\mathcal{M}$ without a given entry (differential privacy), zero-knowledge privacy compares the result of $\mathcal{M}$ to the information which can be derived from the information provided by $agg$, which the designer must define in

order to characterize what information is considered acceptable. Like differential privacy, zero knowledge privacy composes additively when queries are repeated.

Since any zero knowledge private algorithm (regardless of $agg$) can be shown to also be $2\epsilon$-differentially private [7], differential privacy can be regarded as a special case of zero knowledge privacy.

The original paper by Gehrke focuses on the definition of $agg$ as $RS(k(n))$, defined as algorithms which have access to a random sample of $k(n)$ database rows, where $k(n)$ is a function of the sample size $n$. In the trivial case where $agg = RS(n)$, i.e. the simulator has access to all database rows (except $i$), any $\epsilon$-differentially-private algorithm is also $\epsilon$-zero knowledge private with respect to $RS(n)$ [7].

Returning to the previous example of cliques consisting of Republicans and Democrats, it can be shown that a differentially-private algorithm delivering the dominating political preferences of all cliques cannot be zero knowledge private with respect to $RS(k(n))$ where $k(n) \in o(n)$: If $n$ is sufficiently large, then a simulator with access to $o(n)$ samples will not have any information at all about many cliques, making it impossible to calculate statistics for all of them [7].

One simple solution to provide zero knowledge privacy is applying a random sampling step (each answer has probability $s$ of being selected) before the application of a differentially-private mechanism. This leads to a zero knowledge privacy factor $\epsilon_{ZK} = \ln\left(s\left(\frac{2-s}{1-s}\right)\exp(\epsilon) + (1-s)\right)$ with respect to $agg = iidRS(p)$, the class of algorithms performing operations on data after a randomized sampling step with probability $s$ of being selected [28].

In the case of randomized sampling and response as presented in 2.5, the zero-knowledge privacy achieved with respect to $agg = iidRS(s)$ can be calculated on the basis of $\epsilon_{RR}$ as calculated in Equation 2.5, which represents the differential privacy factor achieved without sampling. Applying the relationship from the previous paragraph, we have:

$$\begin{aligned}
\epsilon'_{ZP,RR} &= \ln\left(s\left(\frac{2-s}{1-s}\right)\exp(\epsilon_{RR}) + (1-s)\right) \\
\epsilon'_{ZP,RR} &= \ln\left(s\left(\frac{2-s}{1-s}\right)\left(\max\left(\frac{p+(1-p)q}{(1-p)q}, \frac{p+(1-p)(1-q)}{(1-p)(1-q)}\right) + (1-s)\right)\right)
\end{aligned} \tag{2.10}$$

# 3

# RELATED WORK

A handful of techniques have been developed for differentially-private streaming analysis.

Dwork began examining "differential privacy under continual observation" in 2010 with a proof-of-concept implementing a differentially-private counter [29]. Other methods for differential privacy in streaming have similarly focused on specific statistics; a method proposed by Chan is only capable of calculating the sums of responses [30]. For examining more complex queries about user behavior, these methods are impractical.

RAPPOR, proposed by Erlingsson in 2014, is an algorithm used in Google Chrome to collect user browsing data in a differentially-private manner [21]. It also relies on a distributed privacy model in which user's responses are anonymized before transmission, utilizing randomized response in combination with the hashing of response values using a Bloom filter; the resulting bit strings can then be statistically analyzed to determine the most common reported values. This method is limited to only identifying results with a frequency of greater than around 1%, as any less-frequent results are indistinguishable from statistical noise.

Many of the algorithms which can calculate more complex statistics rely on the presence of trusted nodes, such as the differentially-private mining of frequent graph patterns proposed by Shen [31]. Using a Markov Chain Monte Carlo sampling-based algorithm in combination with the exponential mechanism (a generalized differential privacy method described by Dwork [6]), Shen proposed an algorithm to determine the most frequent subgraphs in a graph dataset. While this application is highly interesting for user behavior analysis, and was indeed tested using a simulated clickstream dataset, it relies on the presence of a trusted curator with access to the full original dataset. As discussed in the introduction, this assumption is unacceptable for many contexts.

The same issue exists with a more flexible system proposed by Friedman, capable of calculating arbitrary statistics. The system bases on the use of nodes which notify the data aggregator when the data collected there reach certain "threshold" values. Laplace perturbations are used at 3 steps in the process to ensure that individual entries cannot (with certainty) trigger or *not* trigger the crossing of a threshold [32]; however, the existence of nodes with access to the original, identifiable data is essential to the concept. SplitX, a system similar to the PrivApprox system discussed below, relies similarly on trusted nodes which mix randomized responses in with accurate responses by data subjects [22]. Another algorithm proposed by Chan also relies on trusted nodes identifying heavy-hitters from accurate data to be transmitted to an untrusted aggregator [33].

The PrivApprox application proposed and implemented by Le Quoc in 2016 uses randomized sampling and response at the client side combined, while hiding information about the source of answers using source rewriting at a series of nodes. Nodes are prevented from associating randomized answers with individual users using a system of One-Time-Pad encryption in which the encrypted values and associated keys are sent separately [8].

Table 3.1: Summary of reviewed differential privacy streaming methods

| Paper | Method | Test Data | Form of Result | Limitations |
|---|---|---|---|---|
| Do Le Quoc et al. [8] | Random sampling and randomized response at client, source rewriting and proxies for anonymity | Taxi trip distances, electricity usage | Answers to SQLite queries in histogram form | Relies on trusted proxies for unlinkability of responses. Possibly vulnerable to longitudinal attacks if respondents are few. |
| Erlingsson et al. [21] | Bloom filter and 2 randomized-response steps at client | Generated normal distribution, Windows process names, and Chrome homepage names | Estimation of appearance rate of heavy-hitters | Complex statistical methods needed to deduce results, only heavy hitters can be identified. Only designed for 1 response per user |
| Chan et al, 2/2012 [30] | Encryption + perturbation on client side, such that reconstruction of sum is possible on analyst side | (No practical implementation) | Sum statistics | Only calculates sum statistics |
| Chan et al, 7/2012 [33] | Misra-Gries algorithm used at trusted nodes, heavy-hitters transmitted to untrusted aggregator | Netflix contest data set, evaluated moving average of movie ratings | Estimation of appearance rate of heavy-hitters | Requires trusted nodes; only identifies heavy hitters |
| Friedman et al [32] | Trusted nodes notify coordinator when statistics reach a certain threshold defined by "safe zones"; differential privacy via Laplace randomization at three steps | Categorization of e-mails/news articles based on counts of keywords | Calculation of arbitrary statistics | Requires synchronization, trusted nodes which calculate statistics from raw data |
| Chen et al. [22] | Additional random answers shuffled among real answers by proxies | Browser history analyzed for most visited websites, numerical browsing activity statistics | Answers to SQLite queries in histogram form | Requires synchronization, trusted nodes |

PrivApprox relies on the distributed storage of raw data on client devices, eliminating the need for a trusted curator and allowing the data to be anonymized before storage on the data collector's servers. The user never sends their data to the analyst; only an answer to an SQLite query, randomized using the randomized response process (see Section 2.5) on each bit of the response. Additional privacy guarantees are ensured by randomly sampling only a portion of the potential respondents, which both improves the factor $\epsilon$ and allows the algorithm to satisfy "zero-knowledge privacy", a stricter privacy standard introduced by Gehrke [7].

The reviewed techniques are summarized in Table 3.1 above.

# 4

# DESIGN

In order to implement a differentially private system for analyzing more complex user behavior patterns, I use the randomized sampling and response procedure introduced by Le Quoc in his technical report on PrivApprox.

In the PrivApprox system, SQLite queries are sent to client devices along with a set of histogram buckets. First, the client devices flip a coin to determine whether to respond at all. If a device responds, it calculates the desired statistic and converts this into a bitstring which indicates the bucket that the answer falls into. Then, it calculates a new bitstring using randomized response for each bit (see 2.5).

This query/answer procedure is advantageous because it is flexible - it can collect any statistic which is capable of being formulated as an SQL query and organized into buckets. This can be accomplished using numerical queries as well as string queries (using regular expressions as buckets, e.g. *.google.com, *.yahoo.com, ...).

In addition, the idea behind randomized sampling and response is relatively easy to explain to laymen compared to more complex mathematical methods. This poses an advantage of acceptance and trust, since users can better understand how their data are anonymized [8].

The published experiments using PrivApprox are thus far limited to simple analyses, where clients are asked for an answer consisting of a single numerical value. My thesis focuses on investigating its potential use for analyzing user behavior in a more nuanced way; for instance, analyzing the frequency of selected event chains rather than the frequency of single events.

To do so, I have implemented a system which simulates queries and responses using randomized sampling and response, while expanding its capabilities to process answers which consist of multiple columns. By extending queries beyond single statistics, it becomes possible to analyze relationships between related data and discover correlations between events.

As test data, I use collections of tweets stored using the Twitter Developer APIs [34]. In order to simulate the distributed set-up of PrivApprox, I select a sample of highly-active users, each of whom symbolizes a client device which processes a query and delivers a randomized answer. Using a series of queries, I provide examples of new analyses and analyze the accuracy of the results, the effect of parameter choices, and the limitations on collecting accurate statistics using these methods.

# 5 IMPLEMENTATION

## 5.1 FRAMEWORKS USED

The entire implementation of my test system was coded in Java 8 with the help of several frameworks detailed below.

### 5.1.1 *SQLite*

In order to store and query the analyzed data, I utilize the relational database management system SQLite. SQLite is a self-contained, embedded database management system, meaning that the database is stored as a normal file in the directory rather than requiring a separate server as in a traditional database management system [35]. Java contains binding functions to SQLite, allowing the database to be easily accessed and changed by the program.

SQLite is used both in the data collection process (to store tweet data) as well as the randomized response process (to execute queries and deliver raw responses which are subsequently randomized and analyzed).

### 5.1.2 *Twitter Developer APIs*

For the collection of Twitter data, I utilize the Streaming and REST APIs of the Twitter Developer kit. The Streaming APIs are used to access newly-arriving data in real-time, while the REST APIs are used to search and query historical tweets [34].

In my implementation, the REST APIs are used in order to search generally for tweets using textual queries and to query the "timelines" (history of tweets) for individual users. These methods are limited by time (6-9 days in the past) and number (up to 3200 tweets) respectively [36]. The Streaming API is used to actively collect new tweets as they are submitted [37]. Both APIs function by accepting textual queries and delivering JSON responses with JSON objects symbolizing users, tweets, and hashtags, among other things.

In order to use the APIs in my Java programs, I use twitter4j, an unofficial Java library which wraps the Twitter API for use in Java [38]. twitter4j configures queries and results as Java objects, simplifying the query design and result processing.

## 5.2 COLLECTION OF TEST DATA

In order to ensure an adequately large data volume and equivalent conditions across tests, a static database was used to test the utility of various queries.

Preliminary experiments, along with the theoretical calculations in Section 2.5.0.4 showed that useful results are associated mainly with two database-dependent factors: a high sample number, and, with respect to each individual bucket, a sufficiently large

proportion of "yes" answers. With the very large amount of data available on Twitter, a large sample number is easy to ensure.

However, ensuring that the buckets will have an adequately large proportion of "yes" answers is not as trivial, especially for string queries. Any non-trivial string term, even if relatively popular, will appear in only a very small proportion of all tweets, resulting in a proportion r which is too small to yield a useful result after randomized response.

To mitigate this problem and allow the useful testing of string queries, I collected a sample of users who could be expected to have similar content. Specifically, users tweeting in support of the American President Donald Trump were identified using the Search API [39] to find all tweets with the hashtag "#MakeAmericaGreatAgain" from the last 6 to 9 days. The users who sent these tweets were then compiled into a list.

This sample group is appropriate since it is large, highly active, and prone to the spread of "viral" hashtags and discussion topics, satisfying the need for a large sample size and the presence of strings which will have a relatively high appearance rate.

After the identification of 10,596 distinct users who had tweeted the hashtag "#MakeAmericaGreatAgain", their histories were collected using the Timeline API [40] to collect their most recent 200 tweets.

The resulting 2,062,801 tweets were parsed and stored in two SQLite tables: "tweets", and "hashtags" (containing the 932,194 hashtags associated with the tweet collection).

In order to facilitate queries which consider sequential events, the additional columns "prev" and "next" were added to the table "tweets", identifying the tweets coming before and after each individual tweet.

## 5.3 ANONYMIZATION

As in the PrivApprox system proposed by Le Quoc [8], my implemented system requires the analyst to set an SQL query, a set of buckets, and the randomization parameters: probabilities s, p, and q, respectively indicating the sampling probability, the probability of giving an honest answer, and the probability of answering "yes" in case of a randomized answer.

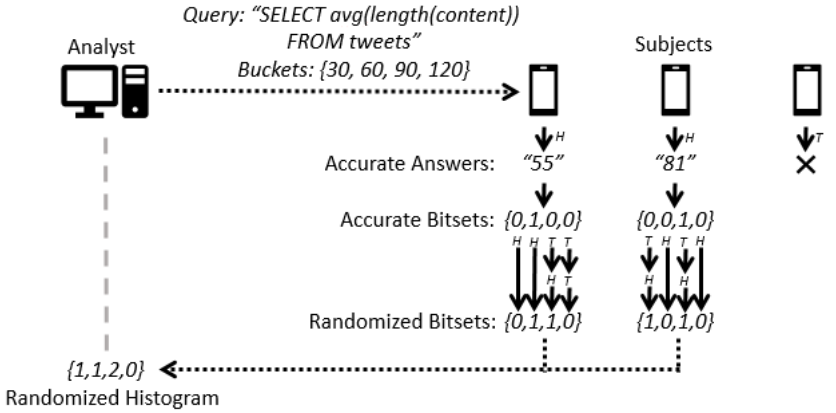


Figure 5.1: A miniature-scale example of the implemented randomized response process. The analyst sends a query and buckets to a set of 3 subjects each convert these to randomized response bitsets, which are then aggregated into a histogram by the analyst. Arrows marked with "H" or "T" represent coinflips resulting in "heads" or "tails" respectively.

The randomization algorithm is run on the SQLite database file after data collection. To convert the query into the final randomized result, Java 8 Streams are used, converting the query into a stream of answers, a stream of bitsets indicating bucket membership, and a stream of randomized bitsets after the application of randomized response. This stream is then collected into a histogram for analysis. This process is described more specifically in the following paragraphs and visually depicted in Figure 5.1.

Depending on the query, a user will submit one or more answers (rows): for instance, the query "SELECT length(content) FROM tweets" will yield one row for each tweet by the user in the database, while the query "SELECT AVG(length(content)) FROM tweets" will only return one answer for each user. For each answer, a coin is flipped with probability $s$ to determine whether to respond at all.

These answers are then compared to the provided bucket values and converted into a bitset (realized as a Boolean array) which communicates the bucket to which the answer belongs. This "accurate bitset" is then converted into a "randomized bitset": for each bit, a coin is flipped with probability $p$ that the bit remains honest. If the bit is not to remain honest, a second coin is flipped, with probability $q$ that the bit reads "true", otherwise "false".

The stream of randomized bitsets is then collected into a histogram by summing the appearances of "true" bits for each position in the bitset. Afterward, the formula in Section 2.5.0.4 is applied to determine the estimated percentage of responses in each bucket.

For comparison purposes, my test implementation includes the accurate results along with the randomized results (wrapped in the class *Results*); however, a real-life implementation would of course remove this information from processing.

For evaluation purposes, the results (including accurate counts, randomized counts, accurate percentages, and percentages estimated from randomized counts) are then written to a CSV file for analysis.

### 5.3.1 *Multi-Column Responses*

In order to allow the algorithm to handle multi-column queries, it is necessary to expand this basic function. The inputs by the user remain fundamentally the same; however, it becomes possible to specify queries which return multiple columns, requiring buckets to be specified for each column.

In order to retain the bitset structure of the responses, it becomes necessary to combine the multiple bucket sets into one bucket set, of which each bucket representing one possible combination of buckets from the original multiple sets. Thus, the buckets $\{a, b, c\}$ and $\{d, e\}$ results in an operational bucket set $\{\{a, d\}, \{a, e\}, \{b, d\}, \{b, e\}, \{c, d\}, \{c, e\}\}$; the operational bucket set can be seen as a Cartesian product of the individual bucket sets.

Answers with $n$ columns are then converted into a corresponding bitset by determining the bucket to which each column of the answer belongs, and then assigning the combination of these buckets to the composite bucket representing that specific combination of $n$ buckets.

To do so, I first calculate a "sub-index" $x_i$ for each column, indicating which bucket the individual column belongs to with regard to its individual bucketset. Then, if $n$ indicates

the number of columns and $b_i$ indicates the number of buckets in the $i$-th column, the overall index of the bit which needs to be set to "1" can be calculated as:

$$\sum_{i=0}^{n}(x_i \cdot \prod_{j=i+1}^{n-1} b_i)$$

For instance, with buckets $\{a, b, c\}$ and $\{d, e\}$ combining to $\{\{a, d\}, \{a, e\}, \{b, d\}, \{b, e\}, \{c, d\}, \{c, e\}\}$, observe the index for an answer belonging to $\{b, e\}$. We have $x_0 = 1$, $x_1 = 1$, and $b_1 = 2$, resulting in $(1 * 2) + (1) = 3$, correctly assigning the positive bit to the third position (with 0-indexing).

After the calculation of the bitsets, the randomization process continues normally as described in the previous section.

## 5.4 EVALUATION METRICS

The two evaluation metrics used to evaluate the results are the differential privacy factor, as well as the coefficient of variation.

The differential privacy factor is calculated using the following formula (see Section 2.5.0.2, Equation 2.6):

$$\epsilon'_{RR} = \ln\left(1 + \left(s\left(\max\left(\frac{p + (1-p)q}{(1-p)q}, \frac{p + (1-p)(1-q)}{(1-p)(1-q)}\right) - 1\right)\right)\right)$$

where $p$ is the probability of responding honestly, $q$ is the probability of responding with "yes" in case of a random answer, and $s$ is the probability of responding at all.

The utility is evaluated using the relative standard deviation, or coefficient of variation $CV = \frac{\sqrt{V(\hat{r})}}{r}$. The value $V(\hat{r})$ can be theoretically calculated as (see Section 2.5.0.4, Equation 2.8):

$$V(\hat{r}) = \frac{(p \cdot r + (1-p) \cdot q)(1 - (p \cdot r + (1-p) \cdot q))}{p^2 \cdot s \cdot N}$$

In addition, I estimate $V(\hat{r})$ empirically for each query by running 100 trials and calculating the unbiased sample variance:

$$V(\hat{r}) = \frac{1}{N-1}\sum_{i=1}^{n}(r_i - \bar{r})^2$$

where $r_i$ is the proportion calculated in each trial, and $\bar{r}$ is the average proportion calculated across all trials.

Since the value $\hat{r}$ is calculated individually for each histogram bucket, the relative standard deviation is also calculated for each bucket in trials.

## 5.5 PARAMETER SELECTION

For a given query with $N$ responses (before the application of randomized sampling), the CV in the calculated statistic for a response with a positive response proportion $r$, as well as the differential privacy level, can be calculated using the formulae referenced in the previous section.

Since we have three adjustable parameters $s$, $p$, and $q$, it makes sense to identify optimal combinations of these parameters; specifically, for a desired differential privacy level $\epsilon'_{RR}$, it should be determined which combination of $s$, $p$, and $q$ yields the lowest variance while satisfying the condition of $\epsilon'_{RR}$-differential privacy.

However, as the coefficient of variance $CV$ is also dependent on the positive answer proportion $r$, $CV$ cannot be uniformly determined for all individual response categories, which each will have a differing value for $r$. As a result, I set a different goal: Choose a value of $s$, $p$, and $q$ which allows the lowest-possible positive answer proportion $r$ to be detected with a coefficient of variation $CV_{max}$ which is determined to be acceptable.

To do so, I apply the following iterative procedure:

1. Set the desired differential privacy factor $\epsilon'_{RR,desired}$.
2. Set the desired maximum $CV_{max}$.
3. Set $r_{min} = 1$.
4. For every possible combination of $s$, $p$, and $q$ (testing values between 0.01 and 1 with an increment of 0.01):
   - Calculate $\epsilon'_{RR}$.
   - If $\epsilon'_{RR} \leqslant \epsilon'_{RR,desired}$: Calculate $CV$ for every possible $r$, starting at 1 and decrementing by 0.001, until $CV \geqslant CV_{max}$. Once the threshhold $CV_{max}$ is exceeded, store the last value of $r$ with $CV \leqslant CV_{max}$.
   - If $r$ is smaller than $r_{min}$, then $r_{min} := r$ and store the current combination of $s$, $p$, and $q$ as $s_{opt}$, $q_{opt}$, and $p_{opt}$.
5. Return $r_{min}$, $s_{opt}$, $q_{opt}$, and $p_{opt}$.

# 6 EVALUATION

## 6.1 SELECTION OF OPTIMAL PARAMETERS

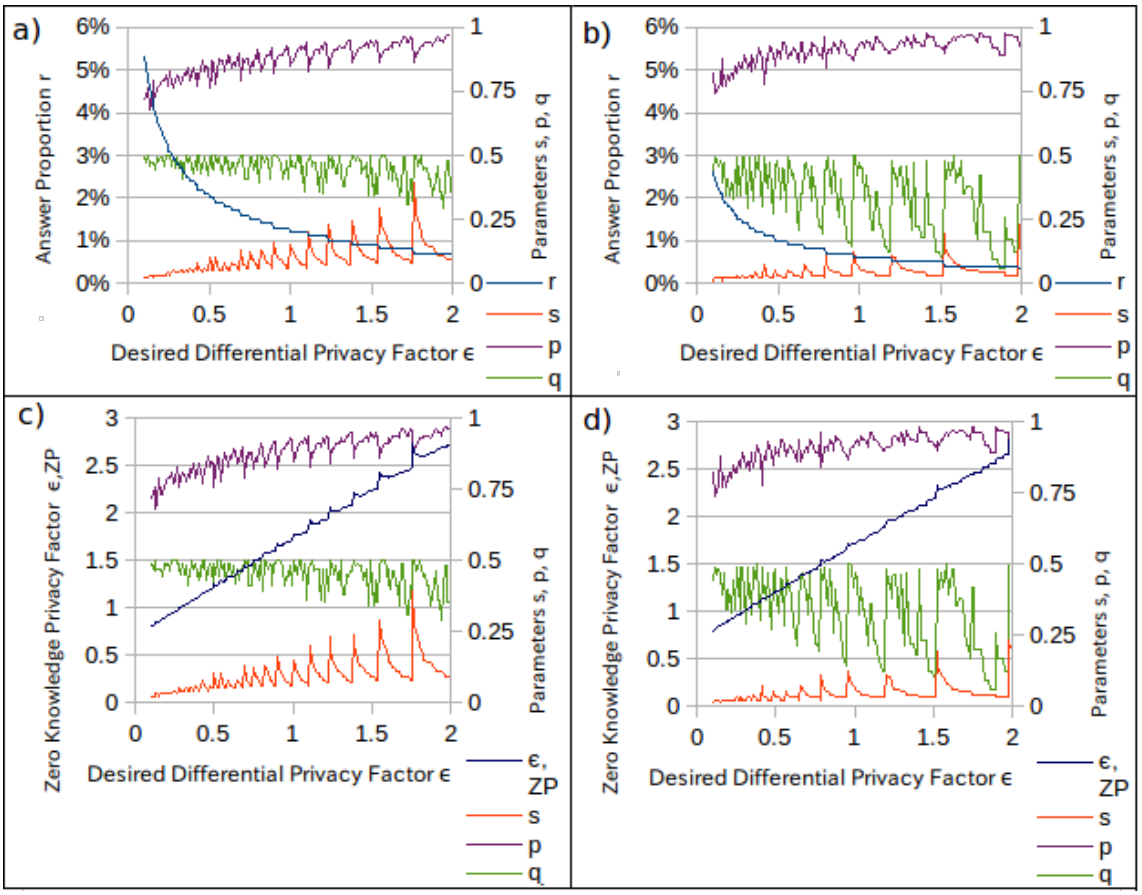


Figure 6.1: Relationship between the desired differential privacy factor $\epsilon$ and the optimal randomization and sampling parameters, along with the minimum answer proportions necessary to achieve a tolerated relative standard deviation a) $\leqslant 5\%$ and b) $\leqslant 10\%$, for a sample size of 2 million. c) and d) depict the zero knowledge privacy factor $\epsilon_{ZP}$ achieved using optimized parameters with a tolerated standard deviation of 5% and 10% respectively.

Using the procedure described in Section 5.5, I calculated the optimal values of parameters s, p, and q for a range of desired differential privacy factors $\epsilon$, along with the corresponding lowest positive answer proportions r yielding a coefficient of variance $\leqslant 5\%$ and $\leqslant 10\%$ (corresponding to relative 95% confidence intervals of $\pm 9.8\%$ and $\pm 19.6\%$ respectively) given a sample size of 2,000,000.

Figure 6.1 shows the resulting optimal parameters s, p, and q for each differential privacy level $\epsilon$, along with the resulting minimum proportion to receive results with the tolerated level of error. As expected, the proportion of positive answers required for

an acceptably accurate answer decreases as $\epsilon$ increases and the corresponding level of privacy decreases. Increasing the error tolerance allows for less-frequent buckets to be accepted. For instance, for $\epsilon = 1$, groups as small as 1.2% may be collected with a 5% tolerated error, while groups as small as 0.6% may be collected with 10% tolerated error.

I additionally examined the relationship between sample size and accuracy by fixing fixed differential privacy factor $\epsilon$ to 0.7 (approximately the value obtained by setting s, p, and q each to 0.5) and repeating the parameter selection procedure for a range of sample sizes from $10^3$ to $10^6$. Table 6.1 shows the degree to which an increased sample size can allow the accurate measurement of small groups; with a sample size of 100,000 and a tolerated coefficient of variation of 5%, we can only accurately measure groups with a frequency of at least 7.88%. Increasing the sample 10-fold to 1 million reduces this requirement to 2.29%; further increasing it to 10 million allows just 0.70% to be measured accurately. (Tolerating a coefficient of variation of 10% reduces the necessary frequency by approximately half in these cases.) This shows that it is essential to have a sufficiently large sample size if the analyst wishes to analyze data in a highly granular fashion.

In addition, I calculated the zero knowledge privacy factor $\epsilon_{ZP}$ achieved (using the parameters optimized with regard to differential privacy). Regardless of the tolerated standard deviation which is selected, $\epsilon_{ZP}$ increases roughly linearly from about 0.8 when $\epsilon = 0.1$ to about 2.7 when $\epsilon = 2.0$.

I also used these results to analyze the benefit achieved from parameter optimization: For the smallest acceptable group frequencies for each sample size (with a coefficient of variation of 5%), I calculated the coefficient of variation which would be observed using a naive parameter selection where p, s, $q = 0.5$ (also yielding $\epsilon \approx 0.7$). The results are depicted in Figure 6.2, which shows that groups with a CV of 5% using optimized parameter selection have a CV up to 5.6% using this case of naive selection. The gain in accuracy from parameter selection is therefore modest; however, it nonetheless has an advantage as a systematic approach for a parameter selection which is an improvement above more arbitrary methods.

## 6.2 STRING QUERY EVALUATION

To test the function of the randomized response algorithm on a simple query of one string value, I queried the text content of all hashtags in the database (*SELECT content FROM hashtags*) and assigned these into buckets indicating the 20 most popular hashtagsy among the user sample. For the first trial, $\epsilon'_{RR}$ was set to 0.7, and the parameters s, p, and q were set to their optimal values of 0.05, 0.88, and 0.37 respectively for a tolerated error of 10% (resulting in zero knowledge privacy $\epsilon'_{ZP,RR} = 1.41$). For the second trial, $\epsilon'_{RR}$ was set to 2.0, with s, p, and q being set to 0.08, 0.94, and 0.2 (resulting in zero knowledge privacy $\epsilon'_{ZP,RR} = 2.72$).

In order to empirically assess the accuracy of the randomized response procedure, I repeated the query 100 times and calculated the empirical variance in the differentially-private result for each bucket.

The results of the first trial with $\epsilon'_{RR} = 0.7$ are summarized in Figure 6.3, charts a) and b). Chart a) demonstrates that more frequent results can be estimated with relatively high reliability; however, less-frequent results are more prone to significant error as a result of randomization.

| Sample size | Proportion with < 5% CV | Proportion with < 10% CV |
|---|---|---|
| $1x10^3$ | 83.27% | 44.50% |
| $5x10^3$ | 39.79% | 18.90% |
| $1x10^4$ | 27.39% | 12.86% |
| $5x10^4$ | 11.37% | 5.37% |
| $1x10^5$ | 7.88% | 3.69% |
| $5x10^5$ | 3.30% | 1.60% |
| $1x10^6$ | 2.29% | 1.10% |
| $5x10^6$ | 0.99% | 0.50% |
| $1x10^7$ | 0.70% | 0.39% |

Table 6.1: Sample sizes and the resulting smallest-possible proportions with acceptable error after parameter optimization.

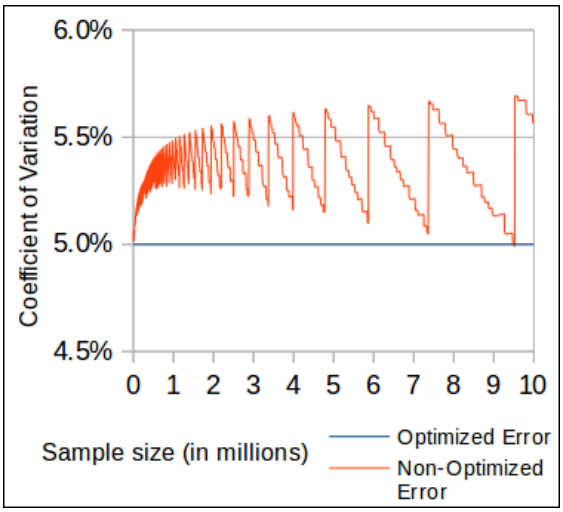


Figure 6.2: Coefficient of variation observed for naive parameter selection (all parameters set to 0.5) vs. optimized parameter selection achieving the same privacy level for various sample sizes, where the examined group has a proportion corresponding to a CV of 5% under optimized conditions.

Chart b) depicts the relationship between the frequency of buckets and utility more clearly. The empirically-observed variation between queries coincides well with the theoretically expected variation calculated using Equation 2.8. The utility of each result increases (i.e. its coefficient of variation decreases) as its frequency increases; results above 1% show more acceptable coefficients of variation $\leqslant 10\%$ (correspoding to a 95% confidence interval of $\leqslant \pm 19.8\%$), while less-frequent results decrease quickly in utility, showing coefficients of variance up to 24% (with 95% confidence interval $\pm 94.1\%$).

The second trial, depicted in charts c) and d), shows a considerably higher accuracy; here, almost all results showed relatively low variability, with a coefficient of variation $\leqslant 10\%$. However, this comes at a cost to the privacy level: The differential privacy factor of 2.0 indicates that any given response may become up to $e^2 \approx 7.4$ times more likely as a result of changing one answer (refer to Definition 2.6.1). While this preserves the fundamental differential privacy characteristic that a given answer cannot be deduced with certainty, an attacker may nonetheless be able to deduce a *probable* answer.

## 6.3 MULTI-COLUMN QUERY EVALUATION

To demonstrate the ability of my system to evaluate queries with multiple columns per response, I queried the length (number of characters) of a given tweet along with the length of the following tweet, using the following query: *SELECT length(a.content), length (b.content) FROM tweets AS a JOIN tweets AS b ON a.next = b.id*. For each column, the 5 buckets [0-50], (50-90], (90-110], (110-130] and (130, 140] were assigned. This results in $5^2 = 25$ composite buckets, each indicating a value range for both columns. The optimal

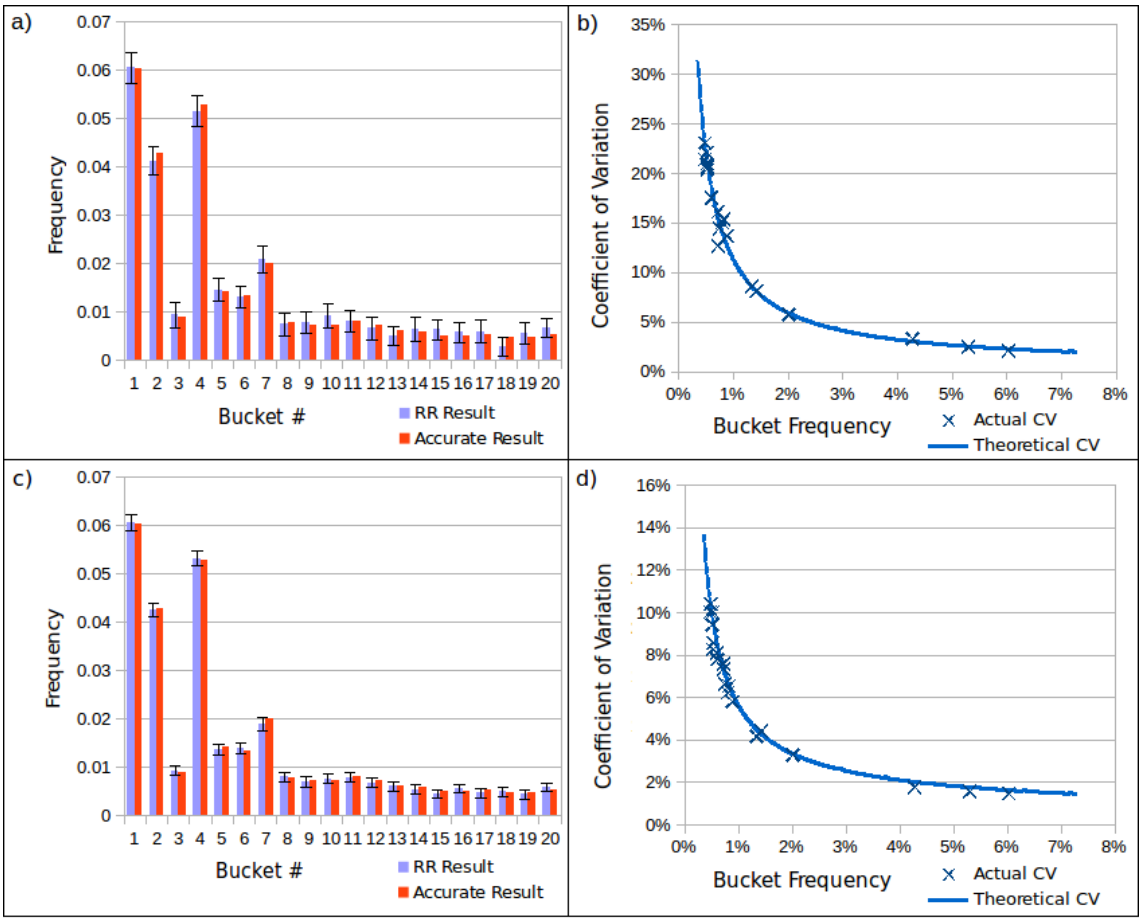


Figure 6.3: a, c) The frequency yielded by one randomized response trial compared to the true frequency for each response bucket. Error bars indicate the 95% confidence interval based on the theoretical standard deviation. b, d) The empirically-observed coefficient of variation for each bucket, compared to the theoretically expected coefficient of variation, in relationship to the true frequency of each bucket. Graphs a) and b) resulted from a trial with $\epsilon = 0.7$; graphs c) and d) from a trial with $\epsilon = 2.0$.

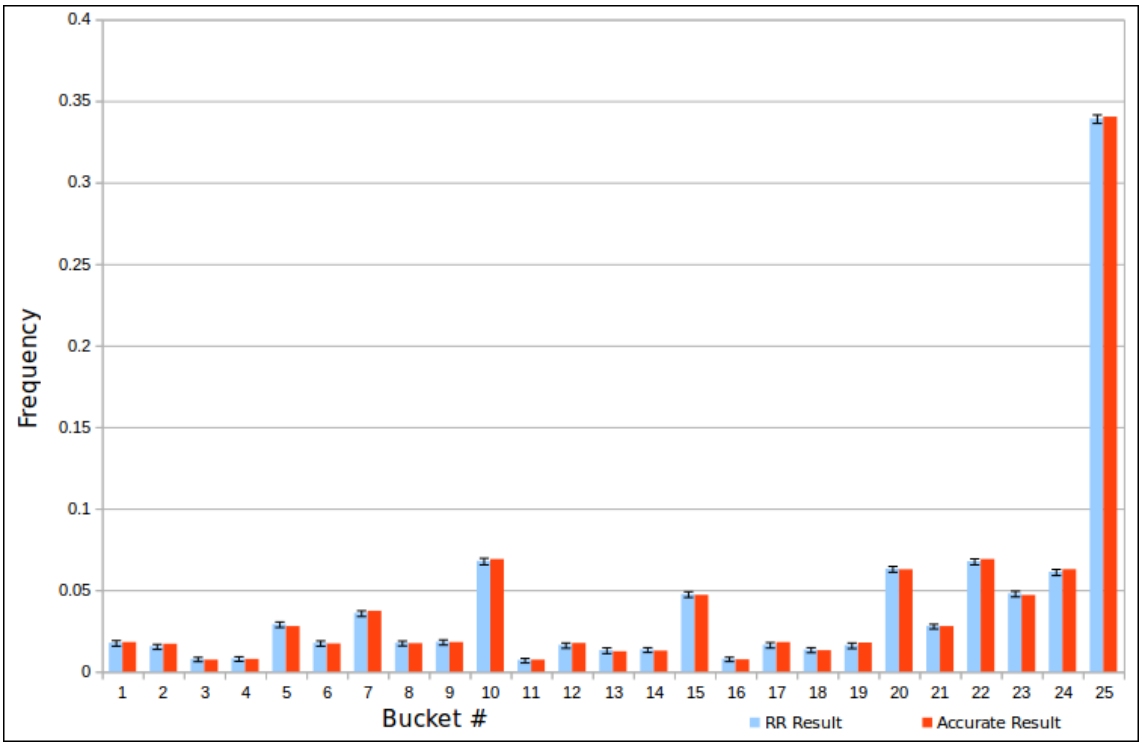


Figure 6.4: The raw results from a multi-column query. Each bucket represents a combination of two value ranges for the length of one tweet and the length of the following tweet. One sample result after randomized sampling and response is compared to the accurate result. The error bars indicate the 95% confidence interval based on the theoretical standard deviation.

parameters for $\epsilon'_{RR} = 0.7$, a tolerated error of 5%, and the response number of 2,052,205 were set to $s = 0.07$, $p = 0.87$, and $q = 0.47$ (resulting in $\epsilon'_{RR,ZP} = 1.42$).

The results are presented in Figure 6.4 in a form similar to Figure 6.3. Here, each bucket represents a specific combination of two value ranges for the "current" and "next" tweet. This graph represents the raw results delivered by the algorithm; however, the depiction is not as useful for interpreting the results in a meaningful way. An analyst would likely not be interested in the frequency of each sequence, but rather in a question such as, "If a user submits a tweet with length $x$, how likely is it that the next tweet has length $y$?"

To better answer this question, I calculated the distribution of lengths in the second tweets of each pair, given that the first tweet falls into a specified value range. This transformation was performed for each individual trial, so that the empirical variance in the end result could be calculated. The results are summarized in Figure 6.5. Here, it can be seen that meaningful results were obtained; for instance, one can deduce that a second tweet with 130 to 140 characters becomes more likely as the length of the first tweet increases. Similarly, a second tweet with less than 50 characters becomes less likely as the length of the first tweet increases.

Figure 6.6 shows that a standard deviation of 10% was not exceeded for any bucket before or after transformation, and that the transformation had no great effect on the variation observed in the buckets.

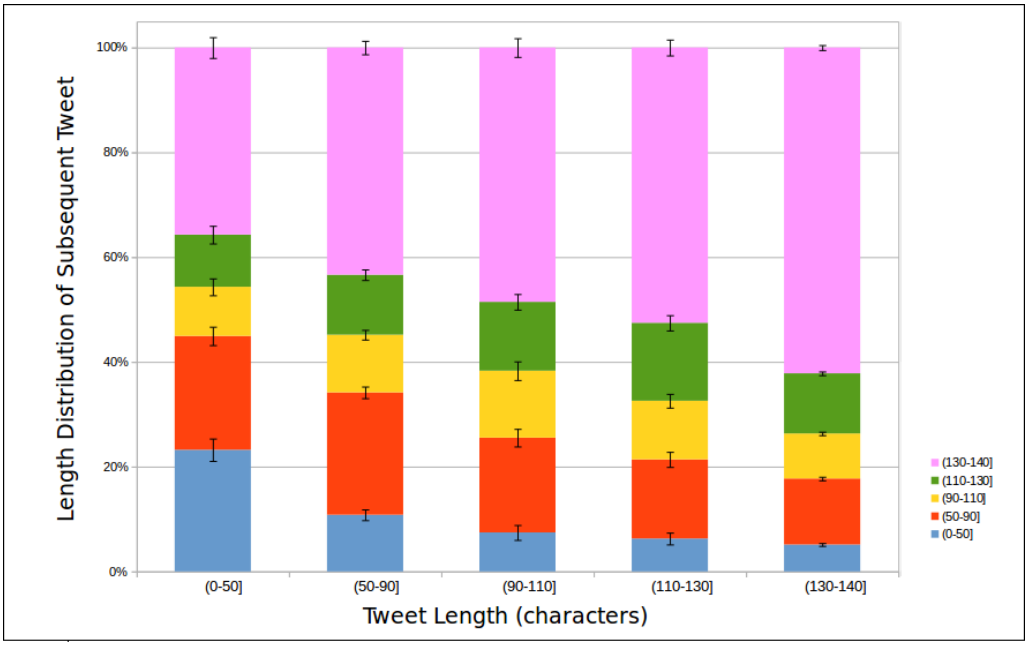


Figure 6.5: The distribution of lengths for a subsequent tweet after a given length for the first tweet. Error bars indicate the 95% confidence interval on the basis of the theoretical standard deviation.

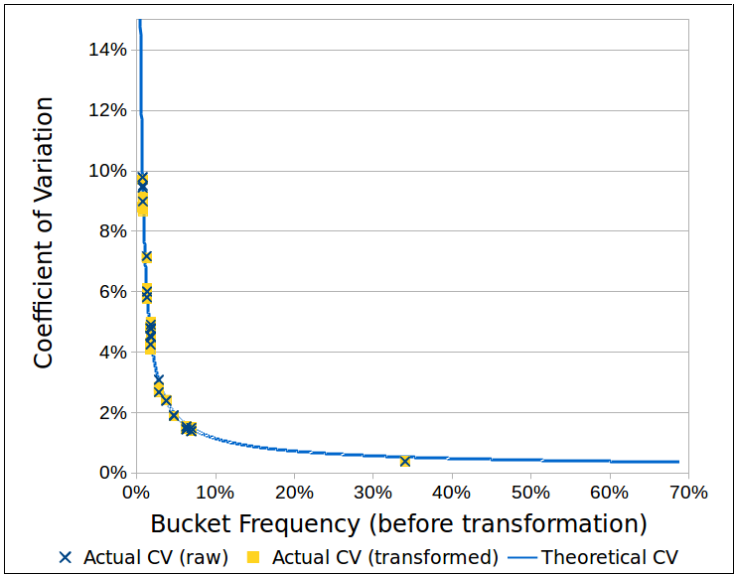


Figure 6.6: The coefficients of variation observed in a multiple-column query, before and after transformation into subgroups, compared to the theoretically-calculated coefficient of variation (before transformation). The transformation performed does not appear to affect the coefficient of variation heavily.

# 7

# CONCLUSION AND FUTURE WORK

## 7.1 SUMMARY

In the course of my thesis, I expanded the existing anonymization technique of randomized sampling and response to be able to process more complex queries consisting of multiple variables. I also derived a theoretical framework to determine the variance introduced by the application of randomized response and determine the limitations the technique.

I characterized the feasibility of anonymization as a function of the size of the surveyed sample, the proportions of the target groups, and the desired differential privacy factor $\epsilon$, and showed that the most accurate results can be obtained by using an iterative technique to determine an optimal combination of the randomization parameters $s$, $p$, and $q$, allowing these parameters to be set objectively rather than arbitrarily (as has generally been done in previous work involving randomized response).

These findings expand the applicability of differential privacy and zero-knowledge privacy in stream processing: Using this technique, it is possible to examine more complex questions relating to sequence of events or correlations between variables. The operator of a website or a smartphone application can use this technique to answer questions such as: How often does a user who clicks on a given advertisement proceed to purchase a product? Which pages are more popular among users of various age groups? Using examples based on publicly-available Twitter data, I showed that results can be obtained in this fashion which are sufficiently accurate and differentially private.

The analysis also revealed certain limitations of this technique. Perhaps the most notable limitation is the fact that the variance of the received answer grows as the total sample size becomes smaller, or as the queried groups become smaller in proportion to the survey group. This fact can make it impossible to derive useful results in situations where the sample size and/or the target group are too small (unless the quality of the differential privacy is sacrificed). However, I took steps to mitigate this problem by establishing a theoretical framework with which analysts can estimate the expected accuracy of their results for given target group proportions, as well as a procedure to minimize variability by optimizing the randomization parameters for a desired privacy level.

## 7.2 FUTURE WORK

Randomized sampling and response may only be used to check the frequency of buckets set by the analyst; it cannot be used to proactively "discover" groups which are not listed in the buckets of the query (beyond the ability to identify the proportion of respondents not belonging to any of the given buckets). However, all differentially-private algorithms which could be found that allow the analysis of arbitrary data types (such as strings) require this input from the analyst [22, 21, 8]. The discovery of such algorithms may be a subject of future work.

Since the differential privacy factor increases additively as a query is repeated multiple times 2.5.1 (and the average results of repeated queries will eventually converge to an accurate result), a real-life implementation of this system would need measures to ensure that the analyst cannot simply repeat a query an arbitrary number of times to circumvent anonymization. In future implementations, it would be plausible to solve this issue using "permanent randomized response", a technique used by Erlingsson in the RAPPOR system by which user devices remember their answers to previous queries and deliver the same randomized response if the query is repeated [21].

A topic of differential privacy which has generally been unsatisfactorily discussed in differential privacy literature is the choice of $\epsilon$. Beyond an abstract understanding of the value of $\epsilon$ (e.g. $\epsilon = 0.7$ indicates that a given true value cannot increase the probability of a given response by a factor greater than 2), little attention is paid in most literature to how this translates to an appropriate choice of $\epsilon$ in the real world. Lee [41] developed a model for determining an acceptable $\epsilon$ on the basis of the underlying distribution of results, and Hsu [42] developed a model based on comparing the economic costs of accuracy to the costs of potential privacy breaches. Future work may focus on applying these or similar models to randomized sampling and response to form a more rigorous process in the selection of $\epsilon$.

# BIBLIOGRAPHY


[1] Clint Boulton. Starwood taps machine learning to dynamically price hotel rooms. *CIO*, 2016.

[2] Christopher Baechle, Ankur Agarwal, and Xingquan Zhu. Big data driven co-occurring evidence discovery in chronic obstructive pulmonary disease patients. *Journal of Big Data*, 4(1):9, 2017.

[3] IDC. Big Data and Business Analytics Revenues Forecast to Reach 150.8 Billion This Year, Led by Banking and Manufacturing Investments, According to IDC. *IDC*, 2017.

[4] Pew Research Center. Public Perceptions of Privacy and Security in the Post-Snowden Era. 2014.

[5] European Parliament. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 2016.

[6] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.*, 9(3&4):211–407, August 2014.

[7] Johannes Gehrke, Edward Lui, and Rafael Pass. Towards Privacy for Social Networks: A Zero-Knowledge Based Definition of Privacy. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 432–449. Springer, 2011.

[8] Do Le Quoc, Martin Beck, Pramod Bhatotia, Ruichuan Chen, Christof Fetzer, and Thorsten Strufe. Privacy Preserving Stream Analytics: The Marriage of Randomized Response and Approximate Computing. *CoRR*, abs/1701.05403, 2017.

[9] Centers for Medicare & Medicaid Services. The Health Insurance Portability and Accountability Act of 1996 (HIPAA), 1996.

[10] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving Data Publishing: A Survey of Recent Developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, June 2010.

[11] Latanya Sweeney. Matching Known Patients to Health Records in Washington State Data. *CoRR*, abs/1307.1370, 2013.

[12] Paul Ohm. Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization. *UCLA Law Review, Vol. 57, p. 1701, 2010*, 2009.

[13] Natalie Newman. Netflix Sued for "Largest Voluntary Privacy Breach To Date". *Proskauer Privacy Law Blog*, 2009.

[14] Alex Hern. 'Anonymous' browsing data can be easily exposed, researchers reveal. *The Guardian*, 2017.

[15] Andreas Bender. Anwendbarkeit von Anonymisierungstechniken im Bereich Big Data [Applicability of anonymization techniques in Big Data]. Master's thesis, Karlsruhe Institute of Technology, 4 2015.

[16] P. Samarati and L. Sweeney. Protecting Privacy when Disclosing Information: k-Anonymity and its Enforcement through Generalization and Suppression. Technical report, 1998.

[17] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy Beyond K-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.

[18] Ninghui Li and Tiancheng Li. t-Closeness: Privacy Beyond k-Anonymity and $\ell$-Diversity. In *In Proc. of IEEE 23rd Int'l Conf. on Data Engineering (ICDE'07*, 2007.

[19] Mehmet Ercan Nergiz, Maurizio Atzori, and Chris Clifton. Hiding the Presence of Individuals from Shared Databases. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 665–676, New York, NY, USA, 2007. ACM.

[20] Matthew Andrews, Gordon T. Wilfong, and Lisa Zhang. Analysis of k-anonymity algorithms for streaming location data. In *2015 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops, Hong Kong, China, April 26 - May 1, 2015*, pages 1–6, 2015.

[21] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1054–1067, New York, NY, USA, 2014. ACM.

[22] Ruichuan Chen, Istemi Ekin Akkus, and Paul Francis. SplitX: High-performance Private Analytics. *SIGCOMM Comput. Commun. Rev.*, 43(4):315–326, August 2013.

[23] Apple previews iOS 10, the biggest iOS release ever. *Apple*, 2016. Accessed: 2017-07-08.

[24] Charlie Cabot. An Introduction to Differential Privacy. *InfoQ*, 2017. https://www.infoq.com/articles/differential-privacy-intro/?utm$_{c}$ampaign = infoq$_{c}$ontentutm$_{s}$ource = infoqutm$_{m}$edium = feedutm$_{t}$erm = global/.Accessed : 2017 − 07 − 08.

[25] Stanley L. Warner. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965. PMID: 12261830.

[26] Ninghui Li, Wahbeh Qardaji, and Dong Su. On Sampling, Anonymization, and Differential Privacy or, K-anonymization Meets Differential Privacy. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, pages 32–33, New York, NY, USA, 2012. ACM.

[27] Donald Bentley. Randomized Response. https://www.dartmouth.edu/~chance/teaching_aids/RResponse/RResponse.html. Accessed: 2017-06-22.

[28] Johannes Gehrke, Michael Hay, Edward Lui, and Rafael Pass. Crowd-blending privacy. In *Advances in Cryptology - Crypto 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 479–496. Springer, 2012.

[29] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential Privacy Under Continual Observation. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 715–724, New York, NY, USA, 2010. ACM.

[30] T. H. Hubert Chan, Elaine Shi, and Dawn Song. *Privacy-Preserving Stream Aggregation with Fault Tolerance*, pages 200–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[31] Entong Shen and Ting Yu. Mining Frequent Graph Patterns with Differential Privacy. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 545–553, New York, NY, USA, 2013. ACM.

[32] J. Sun, R. Zhang, J. Zhang, and Y. Zhang. Pristream: Privacy-preserving distributed stream monitoring of thresholded percentile statistics. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016.

[33] T.-H. Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially Private Continual Monitoring of Heavy Hitters from Distributed Streams. In *Proceedings of the 12th International Conference on Privacy Enhancing Technologies*, PETS'12, pages 140–159, Berlin, Heidelberg, 2012. Springer-Verlag.

[34] Twitter Developer Documentation. https://dev.twitter.com/docs. Accessed: 2017-06-22.

[35] D. Richard Hipp. SQLite. https://www.sqlite.org/. Accessed: 2017-06-28.

[36] Twitter Developer Documentation - REST APIs. https://dev.twitter.com/rest/public. Accessed: 2017-06-28.

[37] Twitter Developer Documentation - Streaming APIs. https://dev.twitter.com/streaming/overview. Accessed: 2017-06-28.

[38] Yusuke Yamamoto. twitter4j. http://twitter4j.org/en/. Accessed: 2017-06-28.

[39] The Search API. https://dev.twitter.com/rest/public/search. Accessed: 2017-07-06.

[40] Working with Timelines. https://dev.twitter.com/rest/public/timelines. Accessed: 2017-07-06.

[41] Jaewoo Lee and Chris Clifton. *How Much Is Enough? Choosing $\epsilon$ for Differential Privacy*, pages 325–340. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[42] Justin Hsu, Marco Gaboardi, Andreas Haeberlen, Sanjeev Khanna, Arjun Narayan, Benjamin C. Pierce, and Aaron Roth. Differential privacy: An economic method for choosing epsilon. *CoRR*, abs/1402.3329, 2014.