Masterarbeit am Institut für Informatik der Freien Universität Berlin,
Arbeitsgruppe ID Management

Differential Privacy: General Survey and Analysis of Practicability in the Context of Machine Learning

Franziska Boenisch

franziska.boenisch@fu-berlin.de
Matrikelnummer: 4821885
Betreuer & 1. Gutachter: Prof. Dr. Marian Margraf
2. Gutachter: Prof. Dr. Tim Landgraf

Berlin, den 8.1.2019

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den 08.01.2019

_____

Franziska Boenisch

Abstract

In recent years with data storage becoming more affordable, every day, increasingly large amounts of data are collected about everyone by different parties. The data collection allows to perform data analyses that help to improve software, track user behavior, recommend products, or even make large advances in the medical field. At the same time, the concern about privacy preservation is growing. Differential Privacy (DP) offers a solution to the potential conflict of interest between the wish for privacy preservation and the need to perform data analyses. Its goal is to allow expressive data analyses on a whole population while achieving a mathematically accurate definition of privacy for the individual.

This thesis aims to provide an understandable introduction to the field of DP and to the mechanisms that can be used to achieve it. It furthermore depicts DP in the context of different privacy preserving data analysis and publication methods. The main focus of the work lies on examining the practicability of DP in the context of data analyses. We, therefore, present implementations of two differentially private linear regression methods and analyze the trade-offs between privacy and accuracy. We found that the adaptation of such a machine learning method to implement DP is non-trivial and that even when applied carefully, privacy always comes at the price of accuracy. On our dataset, even the better-performing differentially private linear regression with a reasonable level of privacy produces a mean squared error twice as high as the normal linear regression on non-privatized data. We, moreover, present two real-world applications of DP, namely Google's RAPPOR algorithm and Apple's implementation of DP in order to analyze the practicability at a large scale and to find possible limitations and drawbacks.

Contents

1 Introduction

Nowadays, digital data about individuals has become increasingly detailed and its collection and curation are facilitated due to constantly improving hardware resources. Thus, massive amounts of data are collected by organizations, research institutions, and governments. This data collection can be extremely useful in many situations. In recent years, especially software development has started to rely heavily on user feedback. When vendors distribute software to a large amount of users, it is often hard to observe how the software is actually being used. By collecting and analyzing usage data, vendors can get a deeper insight and improve their products based on the observations. For research institutions or governments, as other important stakeholders, medical or political datasets usually are of high interest to understand complex relations in populations or particular subgroups.

At the same time, individuals are usually unwilling to share their private data, especially if the data is considered to be sensitive. This impedes the gathering of useful information or the discovery of previously unknown patterns from data which are highly beneficial for the above-mentioned organizations [159]. So while data collection is important for many parties, it can be hard to get the users to collaborate and to convince them that their data is handled properly [88].

Differential Privacy (DP) represents a mathematical field that aims to solve this conflict of interest [62]. It formalizes the notion of privacy and thereby allows to formally prove that certain data collection methods can preserve the users' privacy. It represents a promise made by a data holder to an individual that he will not be affected in any sense by allowing his data to be used in a study or analysis – no matter what other sources of information, studies, or datasets are available. If applied correctly, DP permits to reveal properties of a whole population or group while protecting the privacy of the individuals [58].

To understand the importance of mathematically formalizing privacy, it helps to look at some cases where data was handled without sophisticated techniques. One of the most popular examples of a privacy disclosure occurred in the Netflix competition in 2007 [147]. The company started a competition and offered a 1 million dollars reward to participants who would achieve a 10 % improvement in the Netflix recommender system. The "anonymized" training dataset that was published consisted of viewing data from which Netflix had stripped off identifying informa-

tion such as user names or IDs. This de-identification turned out to be insufficient, as researchers were able to re-identify specific users in the data. The approach they used was to *link* the anonymized Netflix dataset to the non-anonymized dataset of the Internet Movie Database (IMDb), another movie rating portal [66]. This sort of attack is called a *linkage attack*. Other prominent privacy failures due to the lack of sophisticated privacy preserving data release techniques contain the AOL search data leak in 2006 and the release of an "anonymized" medical dataset, in which, linked with publicly available voter records, the governor of Massachusetts could be clearly identified [148].

DP was introduced to protect against arbitrary risks going beyond the above-mentioned examples. It was designed not only to prevent linkage attacks with existent, but also with possible future datasets. Furthermore, it enables us to quantify a privacy loss by a concrete value (making it possible to compare different techniques), to define bounds of tolerated privacy loss, or even to quantify the privacy loss over multiple computations or inside groups, such as families. It is also immune to post-processing. This means that no data analyst can use the output of a DP algorithm to make the algorithm lose its formal privacy level [66].

The goal of this thesis is to offer an understandable introduction to DP and to the mechanisms that can be used to achieve it. Furthermore, the work aims to examining the practicability of DP in the context of data analyses. The following chapter gives an overview on related work on privacy preserving data analysis and publication, on research conducted on DP, and on its application areas. Chapter 3 provides a mathematical formalization of DP, describes the most common mechanisms, and compares their qualities. Chapter 4 presents the experimental part of the thesis. This part consists of the implementation of two differentially private linear regressions in order to investigate the applicability of DP to machine learning algorithms and to evaluate the trade-off between privacy and accurate results. Afterwards, in Chapter 5, some real-world applications of DP are examined, namely Google's *RAPPOR* and Apple's *Count Mean Sketch* and *Sequence Fragment Puzzle* algorithms. The final chapter discusses the qualities and limitations of DP approaches and its usefulness on machine learning tasks. Moreover, it gives an outlook on further research questions.

2 Related Work

This chapter describes related methods used to achieve privacy preserving data publication and analysis. Moreover, it reviews the literature that has been published about DP in general and its application in several domains.

2.1 Other Methods for Privacy Preserving Learning and Data Publication

Privacy preserving data analysis and publication has become a topic of broad interest in recent years. The major goal is to use or publish sensitive data in a way that it can provide useful information over a group of targets while preserving the privacy of individuals. Over the years, different models have been proposed to achieve this goal. It is interesting to review these methods because the basic ideas of some of them can be found in the definition of DP, or because DP solves the issues that these methods raise.

In general, there are two natural models for privacy preserving mechanisms: the *interactive* or *online* and the *non-interactive* or *offline* one. In the non-interactive setting, a trusted entity, the so called *data curator*, collects the data and publishes a *sanitized* version of it. This process is called *anonymization* or *de-identification* in the literature. A sanitization can include several techniques that are depicted in Section 2.1.1.

In the interactive approach, the trusted entity provides an interface through which users can query the data and obtain (possibly noisy) answers [58]. This approach is often used if no information about the queries is known in advance, as this poses severe challenges to the non-interactive model [66]. Any interactive solution yields a non-interactive solution provided the queries are known in advance. In that case, the data curator can simulate an interaction in which these known queries are posed and then publish the results [59].

2.1.1 Privacy Approaches in (Statistical) Databases

The aim of privacy preservation in databases is to limit the usefulness of a database by only making statistical information available and by preventing that any sequence of queries is sufficient to deduce exact information about individuals [165]. In the following, some approaches to privacy preservation in databases are explained and the risks of privacy disclosure they expose are depicted. A broader overview over security control mechanisms for (statistical) databases can be found in [3, 55, 138, 158].

2.1.1.1 Data Perturbation

A widely used technique is *data perturbation* for which actual data values are modified to dissimulate specific confidential individual record information [159]. The database queries are then conducted on the perturbed data.

Perturbation can be achieved by *swapping* individual values between different data records from the same database [40, 46, 47, 54], replacing the original database by a sample from the same distribution sample [95, 102, 127], and adding noise to the values in the database [155, 157].

[70] found that when replacing a database with a sample from the same distribution, if the distribution is smooth enough (when representing a distribution as a density function, the smoothness measures how many times the density function can be differentiated), it is possible to recover the original distribution from summary statistics (see Section 2.1.1.6), thus violating the privacy.

Adding noise to the data offers the advantage that one can not determine the exact values of an individual anymore. The data can still be approximated, but if the noise is sufficiently large, an individual's privacy is effectively protected [110]. However, increased noise makes the variance in the query answers larger, thus, reducing the utility of the query outputs. This is not the only issue with noise addition: if the level of noise is not chosen carefully, it also introduces the problem of a *bias*. [110] showed that such a bias causes the responses to queries to have a tendency to underestimate the true value. With this knowledge, an adversary might find out more about an individual than he should.

2.1.1.2 Output Perturbation

In *output perturbation*, the queries are evaluated on the original data but return a perturbed version of the answers. Techniques include returning the answers only over a sample of the database, so-called *subsampling* [44], rounding the output to a multiple of a specified base [39], and adding random noise to the outputs [11].

Especially the approach of subsampling has proven to disclose individual information. If just a few members of a dataset are represented in a statistic, the background knowledge that a specific individual is in the subsample becomes more powerful. Furthermore, there is the concern that some of these few samples could belong to outliers (data points that are very different from the average one). This would not only decrease the accuracy of the statistics, but in many cases, outliers might be precisely those individuals for whom privacy is most important [66].

2.1.1.3 Access Control

Another possibility to protect privacy in databases is to *control the access* to the data itself. Several methods have been proposed to limit data access based on different principles.

[115] use *cryptographic techniques* to ensure that only authorized users can access the published documents. [19] and [20] describe data access based on a *purpose* specifying the intended use of this data element. Therefore, purpose information is associated with a given data element and only queries for that specific purpose are allowed on the data. [120] depict the *role-based* access principle. In such an approach, every user has a specific role and depending on his role, he has access to different data records. [122] define an access based on *situations*. According to them, a situation is a scenario in which data release is allowed or denied.

All these access control methods have the severe drawback, that they do not allow to release entire databases publicly.

2.1.1.4 Query Restriction

The *query restriction* family includes techniques to *control the overlap* among successive queries [50], to *suppress* data cells with sensitive information in the answers of a query [35, 36, 161], and to *control the query set size* [72, 135, 165]. The latter means

that only those queries which access at least a and at most b database records are allowed on the database.

There exist several privacy issues with the query set size control techniques. Snooping tools, called *trackers*, can be used to learn values for sensitive attributes. This is done by using categories known to describe an individual to query other, potentially sensitive information about that individual [45]. But even without trackers, these techniques do not offer sufficient privacy. If the presence of a record for an individual named A in the dataset D is known, a *differencing attack* can be used to extract information about A without querying it explicitly. Take as an example a medical database containing a field with sensitive data, e.g., whether a person is a smoker (represented by a 1) or not (represented by a 0). When only queries that operate on a dataset with size > 1 and $\leq n$ are allowed, where n is the number of database elements, the following two queries are sufficient to violate the privacy of A and reveal whether he is a smoker by subtracting their results [66].

```
(a) SELECT SUM(isSmoker) FROM D
(b) SELECT SUM(isSmoker) FROM D WHERE name != 'A'
```

2.1.1.5 Query auditing

The idea behind *query auditing* is to decide, dependent on the previous t queries, whether the next query is answered [27, 29, 50]. It can, therefore, be understood as an online version of the query restriction method.

Query auditing is not effective because it can be computationally infeasible or refusing to answer a query itself can be disclosive [66]. [28] showed that pure SUM queries and pure MAX queries can be audited efficiently but that the mixed SUM/MAX problem is NP-hard.

2.1.1.6 Summary Statistics

Another possible method to implement privacy preserving data publication is building *summary statistics*. Instead of releasing specific data records of individuals, only some pre-defined statistics are published. These statistics aim to summarize a set of observations, in order to communicate the largest amount of information as simple as possible.

Summary statistics are not safe, either. They can fail due to, e.g., the above-mentioned *differencing attack* and background knowledge of an attacker. Imagine, for

example, that a statistic over the average height of women in Europe is published. If someone receives the information, that a certain woman is 10 cm smaller than the average, her actual height can be calculated and, thus, her privacy is breached [66].

2.1.1.7 Removal of Specific Identifiers

A further technique to anonymize databases consists of the *removal of specific identifiers* from the data records such as names, IDs, addresses, etc. There is an entire listing from the HIPAA (the 1996 US Health Insurance Portability and Accountability Act) containing 18 categories of personally identifiable information which must be redacted in research data for publication [88].

One of the most common attacks against databases that are sanitized this way is the so-called *linkage attack*. This attack exploits that, even after removal of identifiers, the richness of data can enable *naming*, which is the identification of an individual by a collection of fields that seem, considered on their own, meaningless. Over these combination of fields, the anonymized records of the dataset can be matched against records of a different non-anonymized dataset, as it happened in the Netflix example from Chapter 1 [147]. The fields that were used to de-anonymize the Netflix dataset were the names of three movies watched by the users and the approximate dates when they had watched them [66].

2.1.2 *k*-Anonymity

To deal with the shortcomings of simple data anonymization and to prevent the linkage attacks, Samarati and Sweeney [133] introduced the concept of k-anonymity in 1998. According to their approach, a release of a database is said to fulfill the k-anonymity requirements, if the information for each individual contained in the data cannot be distinguished from at least $k-1$ other individuals whose information also appears in the data.

The linkage attack depicts that, even if all *explicit identifiers* are removed from the database, there exist *quasi-identifiers*, a set of attributes whose release must also be controlled, because otherwise, the sensitive attributes of an individual can be disclosed [147]. k-anonymity uses data generalization and suppression techniques to achieve that every combination of values for the quasi-identifiers can be indistinguishably matched to at least k individuals in one equivalence class.

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	13053	28	Russian	Heart Disease
2	13068	29	American	Heart Disease
3	13068	21	Japanese	Viral Infection
4	13053	23	American	Viral Infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart Disease
7	14850	47	American	Viral Infection
8	14850	49	American	Viral Infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

(a) Real data values.

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	$*$	Heart Disease
2	130**	< 30	$*$	Heart Disease
3	130**	< 30	$*$	Viral Infection
4	130**	< 30	$*$	Viral Infection
5	1485*	≥ 40	$*$	Cancer
6	1485*	≥ 40	$*$	Heart Disease
7	1485*	≥ 40	$*$	Viral Infection
8	1485*	≥ 40	$*$	Viral Infection
9	130**	$3*$	$*$	Cancer
10	130**	$3*$	$*$	Cancer
11	130**	$3*$	$*$	Cancer
12	130**	$3*$	$*$	Cancer

(b) $k = 4$-anonymized database.

Figure 2.1: A fictitious database with hospital data, taken from [109].

In *data generalization*, specific values for different attributes are mapped to broader value ranges to make the data less informative. [132] introduced the concept of *minimal generalization* which is based on the idea to disturb the data not more than necessary, and presented an algorithm to compute it. Figure 2.1 shows a table with real data and its k-anonymized version after the generalization process. In *data suppression*, entire data records are removed from the database. This makes sense for outliers that might otherwise not really fit into the value ranges or render them useless by broadening them too much. Data generalization and suppression can also be used together, and thus, there might exist several solutions to achieve k-anonymization.

Meyerson and Williams [114] showed that finding the optimal k-anonymity for any $k \geq 3$ is NP-hard. However, numerous algorithms have been proposed to approximate k-anonymity [4, 8, 86, 93, 94, 133, 146, 167].

[109, 156, 160] showed that k-anonymity can prevent *identity disclosure*, thus, for no individual it is possible to identify his exact record in the database, but that it does not provide *attribute disclosure*. Attribute disclosure occurs if it is possible, even without finding the exact record of an individual, to reveal his sensitive attributes.

Attribute disclosure on k-anonymized databases can be achieved by two forms of attacks, the *homogeneity attack* and the *background knowledge attack* [109]. A homogeneity attack can be carried out if the sensitive attributes lack diversity. Take as an example Figure 2.1b. All individuals in the third equivalence class have cancer. Thus, finding out the medical issue of an individual of which we know that it is in this equivalence class, is possible even without identifying it directly. Background knowledge attacks rely on partial knowledge about either an individual or a distribution of sensitive and non-sensitive attributes in a population. In Figure 2.1b, if we can use our knowledge about a patient in the database to say with certainty

that he is in the first equivalence class, and if we know that he is unlikely to have a heart disease (because we know his family history or his ethnic background), it is very probable that he is suffering from a viral infection.

2.1.3 ℓ-Diversity

To address these shortcomings of k-anonymity Machanavajjhala et al. [109] introduced ℓ-diversity in 2006. The idea is to prevent homogeneity of sensitive attributes in the equivalence classes.

An anonymized database is said to be ℓ-diverse if in every equivalence class, there exist at least ℓ different values for the sensitive attribute. Figure 2.2 shows an ℓ-diverse version of the hospital dataset from Figure 2.1. To breach any individual's privacy within the table, an adversary would need $\ell - 1$ pieces of background knowledge.

A drawback of ℓ-diversity is that it can be too difficult or unnecessary to achieve. Imagine a scenario with 10 000 data records with a sensitive attribute. Let the attribute be 0 for 99 % of the individuals and 1 for the remaining 1 %. Both possible values have different sensitivities. One would not mind being tested negative, thus, in an equivalence class that only contains 0-entries for the sensitive attribute, ℓ-diversity is unnecessary. But to achieve 2-diversity, there would only exist $10\,000 \cdot 0.01 = 100$ equivalence classes, which would lead to a large information loss [109].

[99] showed that ℓ-diversity is, as well as k-anonymity, insufficient to prevent attribute disclosure as there exist two possible attacks: the *skewness attack* and the *similarity attack*. A skewness attack is possible if the overall distribution of the sensitive attributes is skewed, like in the example with 99 % to 1 %. Imagine that one equivalence class had an equal number of positive and negative records. Then, the database would still be 2-diverse, but the risk of anyone in that class being considered positive would be 50 % instead of 1 % which can harm the individual's privacy. Similarity attacks can be carried out if the sensitive attribute values in an equivalence class are distinct but semantically similar. Take again the hospital example and an equivalence class with sensitive attribute values *lung cancer*, *breast cancer*, and *brain cancer*. These are different values, but all of them refer to a form of cancer, thus, one can learn that an individual in this equivalence class has cancer.

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	1305*	≤ 40	*	Heart Disease
4	1305*	≤ 40	*	Viral Infection
9	1305*	≤ 40	*	Cancer
10	1305*	≤ 40	*	Cancer
5	1485*	> 40	*	Cancer
6	1485*	> 40	*	Heart Disease
7	1485*	> 40	*	Viral Infection
8	1485*	> 40	*	Viral Infection
2	1306*	≤ 40	*	Heart Disease
3	1306*	≤ 40	*	Viral Infection
11	1306*	≤ 40	*	Cancer
12	1306*	≤ 40	*	Cancer

Figure 2.2: An $\ell = 3$-diverse data table, taken from [109].

2.1.4 *t*-Closeness

To overcome the limitations of ℓ-diversity, in 2007, Li et al. [99] introduced the concept of t-closeness. A database is said to fulfill t-closeness, if the distance between the distribution of the sensitive attributes in every equivalence class differs from the distribution of the sensitive attributes in the entire table by at most a given threshold t.

To calculate the difference between the two distributions, the *Earth Mover Distance* [131] is used. This distance captures the minimal amount of work needed to transform one distribution to another by moving distribution mass between them.

In 2010, [98] showed that t-closeness limits the amount of useful information that can be extracted drastically and proposed a more flexible privacy model, called (n, t)-closeness, that allows for more precise analyses within the privacy preserving setting. Other work on t-closeness has been conducted by [142] and [21].

2.1.5 Secure Multiparty Computation

Secure multiparty computation (SMC) is a long-studied problem in cryptography. It has been introduced in 1986 by Yao [164]. It aims to create methods to allow several (untrusted) parties to perform computations over their input data together. The computations should have the property that the parties learn the correct output and nothing else about the other parties' data [13, 25, 52, 76, 136, 163]. Several implementations of SMC prove its feasibility [9, 10, 16, 53, 91, 107, 125].

The idea of SMC is very well applicable to privacy preserving data mining tasks. A good example would be a group of hospitals who want to conduct data analyses on their joint patient data for medical research. The hospitals can, for privacy reasons,

not share their data with each other or with any other party. Other examples include cooperations between intelligence agencies or companies that want to take benefit from joint customer analyses [103].

[91] used SMC to analyze the wage gap between different subgroups in companies in Boston. Therefore, the companies calculated the sums over the salaries for each of the k different (ethnic or gender-related) subgroups and added different large amounts of noise to each of these sums. Then, the k perturbed sums were sent to one server of the Boston University that added up the sums over all companies per subgroup. The k large numbers for the noise were sent to another server that computed k sums, again over all companies per subgroup. At the end, the noisy sums of the salaries were sent to the second server, where the sums of the noise values were subtracted from it. This led to the discovery of the wage gaps without revealing sensitive data from any company.

2.2 Differential Privacy

The field of DP has been introduced by Cynthia Dwork in 2005 in the context of a patent [62]. Her first paper about the subject was published in 2006 [58].

Since then, a lot of work has been conducted to describe the properties and mechanisms of DP and to review the results [57, 59, 64, 65]. The initial work mainly focused on the question of the amount of noise that needs to be added to data to achieve privacy. This led to the introduction of the *Laplace mechanism* [49, 56, 57, 63]. Based on this work, [42, 81] proved lower bounds for the noise-level needed to achieve DP. Related to the question of noise addition, the research on a good choice for the privacy parameter ϵ which defines how well one's privacy is protected (small values of ϵ indicate good privacy) has been conducted by several parties, e.g., [92].

[7, 134] showed that the Laplace mechanism often fails to provide accurate analysis results on numeric data that goes beyond count data because it requires the addition of too large amounts of noise. They also proved that the mechanism also fails to provide privacy over multiple queries, due to so-called *tracker attacks*. In such attacks, an adversary poses several sets of repeating queries and uses the results to estimate the scale and variance of the Laplacian noise. With this, he can restore the real data from the noisy responses. Apart from these drawbacks of the Laplace mechanism, there are datasets which are rendered useless when random noise is added. For those datasets and non-numerical data, the *exponential mechanism* was invented [113].

The most complete work on the foundations of DP is the book "The Algorithmic Foundations of Differential Privacy" written by Dwork and Roth [66]. It defines the notion of DP, motivates its application, presents the mechanisms, their applications, and complexities.

Moreover, the notion of DP has been extended in several directions and for several use cases. Originally, the definition of DP was limited to neighboring databases (see Section 3.1.1 on page 16). [22] investigated the implications of DP on databases with an arbitrary notion of distance. Another weak point of the initial definition of DP was, that it assumes independence of the database records. This might be an unsafe assumption, as in the real world, there exist a lot of dependencies. To counter this problem, [108] introduced the notion of *dependent differential privacy* (DDP), that defines DP for databases with dependences between the records.

[61, 68] used methods from the field of *robust statistics* (a subfield of statistics which attempts to cope with several small errors, e.g., due to rounding errors in measurements) to create new DP methods. The resulting methods were privacy preserving versions of the data scale, median, alpha-trimmed mean, and linear regression coefficients estimations.

In [60], the authors proposed a method for distributed noise generation between multiple parties. Such an approach has the advantage that no trusted data curator is needed anymore. [111] showed limits of distributed DP for two parties.

To face the problem, that, due to the complexity of DP, the creation, or implementation of a lot of DP algorithms are incorrect, [48] developed a counterexample-generator for DP. This generator allows users to test their potential differentially private mechanisms to see, if they really prevent privacy breaches.

2.2.1 Approximate DP

DP gives so high privacy guarantees, that its usage comes at the price of either decreased utility of the query results or increased complexity. This led to the introduction of several weakened definitions of DP.

Random Differential Privacy (RDP), introduced by [80], allows for more accurate query results by replacing some database elements by other elements randomly drawn from the same distribution.

[116] invented the concept of *Computational Differential Privacy* (CDP): whilst traditional ϵ-DP achieves privacy against computationally unbounded adversaries,

the relaxed version CDP provides privacy against efficient, i.e., computationally bounded, adversaries.

The most popular version of approximate DP is (ϵ, δ)-*Differential Privacy*. This approach introduces the parameter δ which represents an upper bound on the probability at which a DP mechanism is allowed to fail. With (ϵ, δ)-DP, one needs smaller databases to achieve the same level of accuracy in analyses as with ϵ-DP. Therefore, for databases of the same size, (ϵ, δ)-DP achieves more accurate results than ϵ-DP [12, 18, 144].

As with ϵ-DP, several approximate DP mechanisms can be combined on the same database. [118] proposed an approximation algorithm to compute the optimal composition of several (ϵ, δ)-DP mechanisms with different privacy levels ϵ_i.

2.2.2 Relation between DP and other Methods

DP adapts several ideas from the basic privacy mechanisms of statistical databases, e.g., the idea of data and output perturbation. But DP offers the great advantage against the simple methods: it proves how much noise is needed to achieve a certain level ϵ of privacy. Also the idea of query restriction is incorporated in ϵ, with ϵ being the maximum tolerated amount of privacy breach.

DP can also be combined with k-anonymity. [141] used k-anonymity as an intermediate step in the generation of datasets for DP, thereby reducing the sensitivity of the data and the amount of noise needed. They showed that with this approach query results became more accurate. [100] showed that when k-anonymization is applied "safely", it leads to (ϵ, δ)-DP where the values of ϵ and δ depend on the choice of k and a value β. This β represents the sample probability, i.e., the probability to include each tuple in the input dataset to a specific sample.

Domingo-Ferrer and Soria-Comas [51, 143] showed that t-closeness and ϵ-DP are closely related. In general, the strategy to create groups of indistinguishable records is a key point for k-anonymity, ℓ-diversity, and t-closeness. The authors proved that when t-closeness is achieved by a strategy they call *bucketization*, i.e., putting several sensitive attributes in so-called *buckets*, t-closeness can yield ϵ-DP when $t = e^{\epsilon}$ [143]. However, in this approach, the utility of the analyses depends heavily on the choice of the buckets because if too many sensitive attributes are put in the same bucket, the accuracy of the results is decreased. Two years later, the same authors introduced *stochastic* t-closeness to account for the fact that DP is stochastic, while traditional t-closeness is deterministic. Based on this concept, they showed that even without bucketization and its related possible loss of utility, $t = e^{\frac{\epsilon}{2}}$ is enough to achieve ϵ-DP [51].

SMC and DP address related but slightly different privacy concerns and application use cases. SMC protocols enable multiple parties to run joint computations on their individual secret inputs such that nothing except from the output of the computation is revealed to each other. DP algorithms address the problem of how to analyze data in a privacy preserving manner such that individual entries are not exposed and yet meaningful and robust data analyses can be performed. In some scenarios, one technology applies, and the other one may not, but sometimes both technologies should be used together to gain the security needed. In [77], the authors present a study on the application of SMC while preserving DP. [145] proposes a system of a SMC protocol for the exponential mechanism (explained in Section 3.2.4). [2] uses DP and SMC for smart metering systems. [126] combines both mechanisms on time series data.

2.2.3 Application of Differential Privacy

DP has been employed in several application areas so far. See [88] for a complete survey on the application of DP with different machine learning algorithms and [87] for specific use cases of DP in big data.

Dwork stated already in the early years of DP that only with noisy sums (which are easy to generate), a lot of data mining tasks, as finding decision trees [129], clustering [14], learning association rules, etc. were possible [56]. But the authors of [74] warned that a naive utilization of DP to construct privacy preserving data mining algorithms can lead to inferior data mining results. This is when more noise than needed is introduced which makes the mining results inaccurate. They then proposed an algorithm which is a DP adaptation of the ID3 algorithm [123] to build decision trees. This idea was later extended by [73] to the construction of random decision forests.

In [1, 137], the authors presented their results of the application of DP for deep learning. Other application areas of DP are personalized online advertising [105], aggregation of distributed time series data [126], private record matching (identifying records between two different parties in their respective databases) [85], and frequent itemset mining (in sets of items, finding the items that appear at least a certain number of times) [101].

Furthermore, techniques from machine learning can also be employed to improve the results of DP queries, as shown by [67]. They used boosting on DP queries and showed that this achieved more accurate summary statistics.

3 Differential Privacy

After the general introduction to the field of privacy preservation in data analysis and publishing, and to DP and its usefulness, this chapter now examines DP from a formal mathematical view. Therefore, first, the terms used in the context are introduced. Then, based on a concrete example, the intuition into the mathematical formulation of DP is given. Afterwards, ϵ-DP and (ϵ, δ)-DP are formally defined and the meaning of the parameters ϵ and δ is explained. Subsequently, an extension of the definition of DP to group privacy and multiple computations on the same dataset is presented. At the end of the chapter, the three basic mechanisms used to achieve DP are depicted and compared against each other.

3.1 Mathematical Formalization of Differential Privacy

For the formalization of DP, the following terms are important. Their definition is adapted from [66].

3.1.1 Notation and Terms

Data curator. A data curator manages the collected data throughout its life cycle. This can include data sanitization, annotation, publication, and presentation. The goal is to ensure that the data can be reliably reused and preserved. In DP, the data curator is responsible for ensuring that the privacy of no individual represented in the data can be violated.

Adversary. The adversary represents a data analyst that is interested in finding out (sensitive) information about the individuals in the dataset. In the context of DP even the legitimate user of the database is referred to as an adversary, as his analyses can damage the individuals' privacy.

Database. We can think of a database D as a multiset with entries from a finite universe $\mathcal{X}$. It is often convenient to represent a database by its histogram $D \in \mathbb{N}^{|\mathcal{X}|}$,

in which each entry d_i represents how often an element $k_i \in \mathcal{K}$ with $1 \leq i \leq |\mathcal{K}|$ appears in D. Here, $\mathbb{N} = \{0, 1, 2, \ldots\}$. In the following, we will use the words *database* and *dataset* interchangeably.

Take as an example a database containing the surnames of the five members of a fictitious group. We can represent the database by its rows as follows.

Müller
Meyer
Schmidt
Müller
Mayer

Here, the universe $\mathcal{X}$ consists of all possible surnames. The collection of records has the form `{"Müller", "Meyer", "Schmidt", "Müller", "Mayer"}`. A histogram notation could be `[2,1,1,1,0,...,0]`, or for better understandability `{"Müller": 2, "Meyer": 1, "Schmidt": 1, "Mayer": 1}` and a zero count for any other surname from the universe.

ℓ_1 norm. The ℓ_1 norm of a database D is denoted by $\|D\|_1$. It measures the *size* of D, i.e., the number of records it contains and can be defined as

$$\|D\|_1 := \sum_{i=1}^{|\mathcal{X}|} |d_i|.$$

ℓ_1 distance. The ℓ_1 distance between two databases D_1 and D_2 is $\|D_1 - D_2\|_1$. It measures how many records *differ* between both databases and can be defined as

$$\|D_1 - D_2\|_1 := \left(\frac{\sum_{i=1}^{|\mathcal{X}|} |d_{1,i} - d_{2,i}|}{2} \right).$$

Neighboring databases. Two databases D_1 and D_2 are called neighboring, if they differ only in at most one element. This can be expressed by

$$\|D_1 - D_2\|_1 \leq 1. \tag{3.1}$$

Mechanism. DP itself is only an abstract concept. The realization of its mathematical guarantees is done by a concrete mechanism, an algorithm that releases statistical information about a dataset.

3.1.2 Motivating Example

As stated in Chapter 1, DP addresses the paradox of learning nothing about an individual while learning useful information about a population. Assume that the information whether an individual is a smoker or not is sensitive. (This could become the case, for example, if health insurances decide to raise the policy prices for smokers.) If we want to find out the percentage of smokers in a population, we need to be able to guarantee that no individual who provides his data for the database will be harmed directly through his participation [66].

To see how DP achieves this goal, we can consider the following game adapted from [168]. The game aims to find whether a given analysis method proposed by a data curator provides DP for individuals whose data is treated with the method. To do so, an adversary creates two neighboring datasets about some individuals' smoking habits. Then, the data curator applies his method to one of the datasets without the adversary knowing to which one. The data curator passes the result of his analysis to the adversary who tries to guess from the result on which of the datasets the method was executed. If he can tell the results on different datasets apart and guess correctly, the privacy of the individual that is different in both datasets is breached because the difference in both results allows conclusion about his smoking habits.

We can describe the game more formally: A data curator implements a summary statistic called $\mathcal{K}$, $\mathcal{K} : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ for any k. This summary statistic can, for example, be the percentage of smokers in a database. An adversary proposes two neighboring databases D_1 and D_2, containing information about the participants' smoking habits, and a set S that represents a subset of all possible values that $\mathcal{K}$ can return. As a simplification, we can assume that S is an interval.

In that very example, let D_1 and D_2 both represent the results from a survey where the participants have been asked whether they smoke (1) or not (0). Let both databases be chosen by the adversary with size $n = 100$ and let them have the following form:

- $D_1 = \{0\colon 100, 1\colon 0\}$ (100 zeros) and
- $D_2 = \{0\colon 99, 1\colon 1\}$ (1 one and 99 zeros).

The adversary also gets to pick the boundaries of $S = [T, 1]$ such that $\mathcal{K}(D_i) \geq T$ if and only if $i = 2$. This means that the adversary knows that $\mathcal{K}$ has evaluated database D_2.

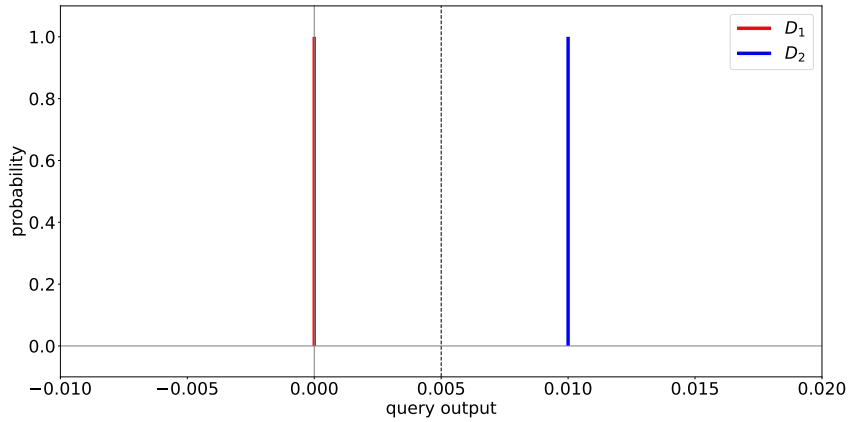


Figure 3.1: In the deterministic case, the probabilities for $\mathcal{K}$ to output 0 on D_1 and 0.01 on D_2 are both 1. If the adversary chooses any $0 < T < 0.01$, he can be sure, that when $\mathcal{K}$ outputs a value $\geq T$, $\mathcal{K}$ was evaluated on D_2. In the example $T = 0.005$ is represented by the dotted line.

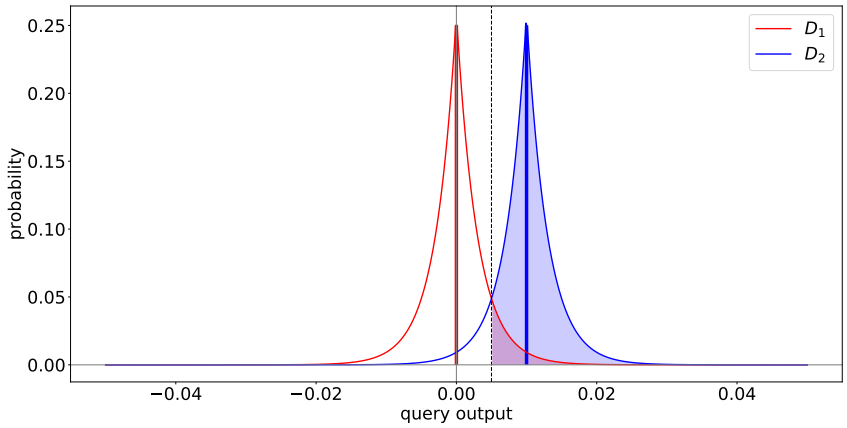
The adversary's goal is to use $\mathcal{K}$ to tell apart D_1 and D_2, resulting in a loss of privacy. The data curator has two opposing goals: First, he wants to pick a $\mathcal{K}$ such that $\mathcal{K}(D_1)$ and $\mathcal{K}(D_2)$ are so close that the adversary cannot find a reliable T. This preserves DP. Second, he also wants $\mathcal{K}$ to be a good estimate of the expectation to perform useful analyses.

The deterministic case

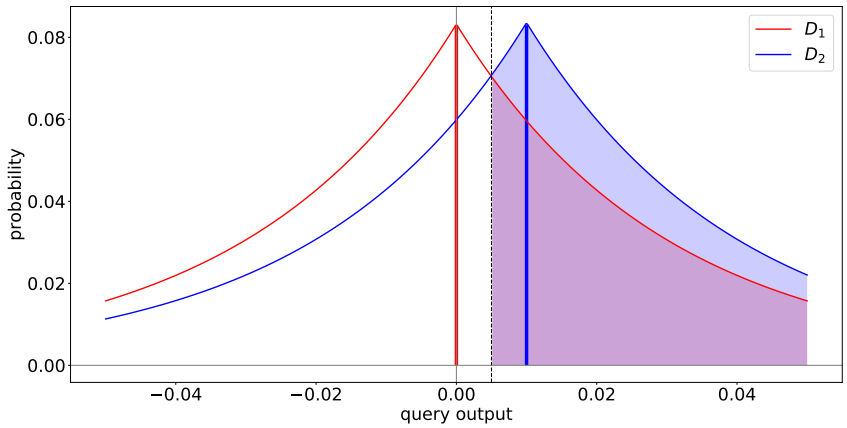
If $\mathcal{K}$ is chosen to be a deterministic summary statistic, e.g., the *mean*-value over the dataset, it would return $\mathcal{K}(D_1) = 0$ and $\mathcal{K}(D_2) = 0.01$. In this case, if the adversary chose $T = 0.005$, he could reliably tell apart both datasets in each round of the game. This setting is depicted in Figure 3.1.

The non-deterministic case

We understand from this example that in a deterministic setting, no privacy can be achieved. Hence, to achieve DP, a controlled amount of random noise needs to be added to the results of the queries [66]. This noise can hide the differences between the datasets. One possible distribution that the noise can be added from is the *Laplace distribution*, which is explained further in Section 3.2.3.



(a) Noise from a Laplace distribution with scale $b = 0.003$.



(b) Noise from a Laplace distribution with scale $b = 0.03$.

Figure 3.2: Different amounts of noise change the probability that the adversary will mistake D_1 for D_2. The level of privacy is, thus, given by the ratio of the values for D_1 and D_2 at each point.

Figure 3.2 shows the probability distributions for possible query outputs over both datasets after adding different amounts of noise from the Laplace distribution to the original query outputs. The red shaded area represents the probability that $\mathcal{K}(D_1)$ will return a value greater than T. This is the chance that the adversary will mistake D_1 for D_2. In Figure 3.2a, the probability for $\mathcal{K}(D_2) \geq T$ is still a lot greater than the probability of $\mathcal{K}(D_1) \geq T$. A larger amount of noise, as shown in Figure 3.2b, can lead to both shaded areas having nearly the same size.

From the visualization, it becomes intuitive that the amount of privacy of the summary statistic depends on the ratio between the probabilities for D_1 and D_2. At

every point, the probability of obtaining a certain result with $\mathcal{K}$ when the database is D_1 should be *close* to the probability of obtaining this result when the database is D_2. The closeness can be expressed by a multiplicative factor close to 1, e.g., $(1+\epsilon)$. See Equation (3.5) for a mathematical formulation of this idea. The parameter ϵ is, thus, an upper bound for the loss of privacy by $\mathcal{K}$.

3.1.3 ϵ-Differential Privacy

The previous example gave an intuition about why adding noise is needed and what the privacy parameter ϵ expresses. This leads to the formal definition of ϵ-DP.

Definition 3.1 (ϵ-Differential Privacy, adapted from [58, Definition])**.** A randomized algorithm $\mathcal{K}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ gives ϵ-DP, if for all neighboring databases $D_1, D_2 \in \mathbb{N}^{|\mathcal{X}|}$, and all $S \subseteq \mathrm{Im}(\mathcal{K})$

$$\Pr[\mathcal{K}(D_1) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{K}(D_2) \in S]. \tag{3.2}$$

Since D_1 and D_2 can be exchanged, this definition directly implies the following lower bound

$$\Pr[\mathcal{K}(D_1) \in S] \geq e^{-\epsilon} \cdot \Pr[\mathcal{K}(D_2) \in S]. \tag{3.3}$$

Combining Equation (3.2) and Equation (3.3) we obtain the constraint

$$-\epsilon \leq \log\left(\frac{\Pr[\mathcal{K}(D_1) \in S]}{\Pr[\mathcal{K}(D_2) \in S]}\right) \leq \epsilon. \tag{3.4}$$

Here and in the following, the notion $\log x$ is used to express $\ln(x)$.

[134] opt for a slightly different formulation, based on the idea of the multiplicative factor $(1+\epsilon)$ that we saw in the motivating example. They define a mechanism to be ϵ-differentially private if

$$\Pr[\mathcal{K}(D_1) \in S] \leq (1+\epsilon) \cdot \Pr[\mathcal{K}(D_2) \in S] \tag{3.5}$$

and call the other definition e^{ϵ}-DP. When ϵ is small, the difference between $(1+\epsilon)$ and e^{ϵ} is negligible. But already with $\epsilon \geq 0.5$, e^{ϵ} is by at least 10 % larger. When ϵ increases, the values diverge considerably. However, e^{ϵ} is the commonly used factor. [7] argue that working with e^{ϵ} has the advantage, that it corresponds to a distribution curve already well known, the Laplace distribution curve. According

to them, this curve has the qualities needed for DP: when the curve is shifted over a certain amount, the ratio of probabilities for the original and shifted curve stay within a predesignated boundary.

Working with e^ϵ, and thus logarithmic probabilities, has also other advantages in the practical application with computers:

1. *Computation Speed:* The product of two probabilities corresponds to an addition in logarithmic space and multiplication is computationally more expensive than addition.

2. *Accuracy:* Using logarithmic probabilities improves numerical stability when probabilities are very small, because of the way in which computers approximate real numbers. With normal probabilities they produce more rounding errors.

3. *Simplicity:* Many probability distributions, especially the ones from which the random noise is drawn, have exponential form. Taking the logarithm of these distributions eliminates the exponential function, making it possible to calculate with the exponent only.

ϵ is often also called the *privacy budget*, as it limits the loss of privacy that an individual or a group is allowed to accumulate to protect their privacy. Equation (3.4) shows clearly that small values for ϵ correspond to good privacy. Especially with $\epsilon = 0$, perfect secrecy is obtained. A good example for a mechanism exhibiting the perfect secrecy property is the 1-*bit one-time pad mechanism*. The input to the mechanism is a bit $b \in \{0, 1\}$. The mechanism chooses a new random bit $p \in \{0, 1\}$ and outputs $(b + p) \bmod 2$.

Theorem 3.2 (adapted from [64, Claim 1])**.** *The 1-bit one-time pad mechanism yields perfect secrecy.*

Proof (taken from [64]). The universe of the databases is $\{0, 1\}$, as well as the image of the mechanism. By exhaustively checking all the possibilities, we see that $\frac{1}{2} = \Pr[\mathcal{K}(0) = 0] = \Pr[\mathcal{K}(0) = 1] = \Pr[\mathcal{K}(1) = 0] = \Pr[\mathcal{K}(1) = 1]$. Hence, plugging this result of the mechanism into Definition 3.1 yields

$$\forall D_1, D_2, \forall S \subseteq \mathrm{Im}(\mathcal{K}) : \frac{\Pr[\mathcal{K}(D_1) \in S]}{\Pr[\mathcal{K}(D_2) \in S]} = 1 = e^\epsilon \Rightarrow \epsilon = 0.$$

□

This perfect secrecy means that the output of the mechanism reveals no information about the database. All perfectly private mechanisms are differentially private, but

the class of differentially private mechanisms contains mechanisms that do not yield perfect privacy. Instead, they provide some non-trivial information about the database. We need that if we want to do (privacy preserving) analyses on the data [64].

3.1.4 (ϵ, δ)-Differential Privacy

ϵ-DP has a high privacy requirement for mechanisms. But since adding too much noise to the original data can limit the amount of information drastically, several weakened versions of DP have been proposed. One of the most popular versions is (ϵ, δ)-DP [66].

Definition 3.3 ((ϵ, δ)-Differential Privacy, adapted from [66, Definition 2.4])**.** A randomized algorithm $\mathcal{K}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ provides (ϵ, δ)-DP, if for all neighboring databases $D_1, D_2 \in \mathbb{N}^{|\mathcal{X}|}$ and all $S \subseteq \mathrm{Im}(\mathcal{K})$

$$\Pr[\mathcal{K}(D_1) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{K}(D_2) \in S] + \delta. \tag{3.6}$$

(ϵ, δ)-DP allows to violate the requirements of ϵ-DP to a certain additive degree fixed by δ. If we look at (ϵ, δ)-DP from a probabilistic view, we see, that δ represents the probability that the mechanism's output varies more than the factor e^{ϵ} between two neighboring datasets [88]. Or more formally, the absolute value of privacy loss will be bounded by ϵ with probability at least $(1 - \delta)$ [66]. If $\delta = 0$, (ϵ, δ)-DP is equivalent to ϵ-DP [66]. Therefore, in the following we are going to use ϵ-DP instead of $(\epsilon, 0)$-DP.

This shows that δ should be small. Intuitively, that makes sense, because, with a large δ, even if $\epsilon = 0$ (perfect secrecy), a mechanism that is $(0, \delta)$-differentially private, would breach privacy with high probability. A common heuristic to choose δ for a database with n records is $\delta \in o(1/n)$. This is because with an (ϵ, δ)-DP mechanism, the privacy for each record in the database is given away with probability δ. By expectation, the algorithm releases $n\delta$ records, thus, δ must be smaller than $1/n$ [88]. But still, [42] showed that (ϵ, δ)-DP is much weaker than $(\epsilon, 0)$-DP, even when δ is very small in relation to the size n of the database.

3.1.5 Group Privacy

The notion of DP can also be extended to groups. But before discussing group privacy, it is necessary, to define the notion of a *group*. Dwork [66] thinks of groups mainly as families or related individuals. According to [150], every collection of

individuals, such as political collectives, ethnic groups, but also groupings created by algorithms, can represent a group. Taylor [106] goes even further and suggests that it is best to see groups as nothing pre-defined that exists naturally, but as something that is created by a data analyst. The need for privacy in groups emerges with the fact that big data analyses are performed on large and undefined groups to identify patterns and behaviors. A breach in privacy occurs if an analysis reveals some sensitive property about the group even when no single individual can be identified [150].

Theorem 3.4 (adapted from [66, Theorem 2.2])**.** *Any ϵ-differentially private mechanism $\mathcal{K}$ is $(k\epsilon)$-differential private for groups of size k. Thus, it is for all databases D_1 and D_2 that differ in at most k elements and all ranges $S \subseteq \mathrm{Im}(\mathcal{K})$*

$$\Pr[\mathcal{K}(D_1) \in S] \leq e^{k\epsilon} \cdot \Pr[\mathcal{K}(D_2) \in S].$$

Proof (not given in [66]). Let D_1 and D_2 be two databases with $\|D_1 - D_2\|_1 \leq k$. Now, we can introduce databases $Z_0, Z_1, ..., Z_k$ with $Z_0 = D_1$ and $Z_k = D_2$. For each database Z_i it holds that $\| Z_i - Z_{i+1} \|_1 \leq 1$. With Definition 3.1 applied to them, and with fixing any event $s \in S$ from the ranges, we get

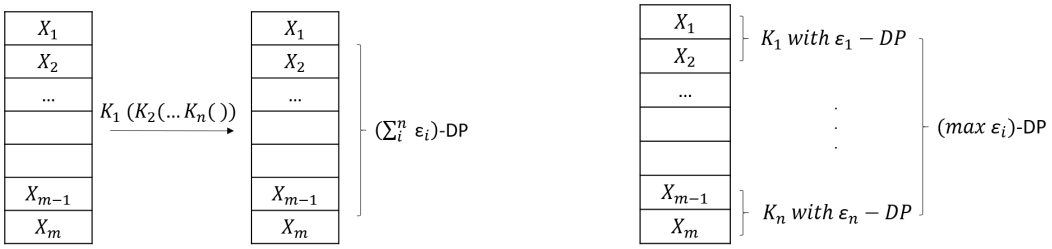
$$\begin{aligned}
\Pr[\mathcal{K}(D_1) = s] &= \Pr[\mathcal{K}(Z_0) = s] \\
&\leq e^{\epsilon} \Pr[\mathcal{K}(Z_1) = s] \\
&\leq e^{\epsilon} e^{\epsilon} \Pr[\mathcal{K}(Z_2) = s] = e^{2\epsilon} \Pr[\mathcal{K}(Z_2) = s] \\
&\vdots \\
&\leq e^{k\epsilon} \Pr[\mathcal{K}(Z_k) = s] = e^{k\epsilon} \Pr[\mathcal{K}(D_2) = s].
\end{aligned}$$

□

This also shows that, if the group becomes larger, the privacy guarantee deteriorates. However, this is what we want. If we replace an entire surveyed population with a different group of respondents, we should get different answers. In (ϵ, δ)-DP, the approximation term δ degrades a lot when dealing with groups and we can only achieve $(k\epsilon, k\epsilon^{(k-1)\epsilon}\delta)$-DP [66]. For further explanation about (ϵ, δ)-DP in groups see [144].

3.1.6 Composition of DP-mechanisms

Any approach to privacy should address the issues of *composition*. Composition is the execution of several queries on the same dataset. These queries can be independent, dependent, or even operate on each other's outputs. DP can capture



(a) Sequential Composition. (b) Parallel Composition

Figure 3.3: For sequential composition, a series of computations with DP mechanisms is executed on the same database, one after the other. In parallel composition, the mechanisms are applied to disjoint subsets of the database in a parallel manner.

composition within one and within several different DP mechanisms. Certainly, the parameters ϵ and δ degrade. There are two forms of composition, *sequential* and *parallel composition* [112]. Figure 3.3 visualizes the ideas behind both approaches.

3.1.6.1 Sequential Composition

Having multiple computations each of which provide DP, it will be shown that they also provide DP if they are composed sequentially. This does not only hold for computations that are run independently, but also when subsequent computations use the results of the preceding computations [112].

Theorem 3.5 (DP for sequential composition, adapted [66, Theorem 3.14])**.** *Let $\mathcal{K}_1$ with $\mathrm{Im}(\mathcal{K}_1) = S_1$ be an ϵ_1-differentially private algorithm and let $\mathrm{Im}(\mathcal{K}_2) = S_2$ be an ϵ_2-differentially private algorithm. Then, their combination $\mathrm{Im}(\mathcal{K}_{1,2}) = S_1 \times S_2$ is $(\epsilon_1 + \epsilon_2)$-differentially private.*

Proof (taken from [66]). Let D_1 and D_2 be neighboring databases. Fix any $(s_1, s_2) \in S_1 \times S_2$, then

$$\begin{aligned}\frac{\Pr[\mathcal{K}_{1,2}(D_1) = (s_1, s_2)]}{\Pr[\mathcal{K}_{1,2}(D_2) = (s_1, s_2)]} &= \frac{\Pr[\mathcal{K}_1(D_1) = s_1]\Pr[\mathcal{K}_2(D_1) = s_2]}{\Pr[\mathcal{K}_1(D_2) = s_1]\Pr[\mathcal{K}_2(D_2) = s_2]} \\ &= \left(\frac{\Pr[\mathcal{K}_1(D_1) = s_1]}{\Pr[\mathcal{K}_1(D_2) = s_1]}\right)\left(\frac{\Pr[\mathcal{K}_2(D_1) = s_2]}{\Pr[\mathcal{K}_2(D_2) = s_2]}\right) \\ &\leq e^{\epsilon_1} e^{\epsilon_2} = e^{\epsilon_1 + \epsilon_2}.\end{aligned}$$

□

Repetitive application of this theorem leads to the following corollary.

Corollary 3.6 (adapted from [66, Corollary 3.15]). *Let $\mathcal{K}_i$ with $\mathrm{Im}(\mathcal{K}_i) = S_i$ be an ϵ_i-differentially private algorithm for $i \in [k]$, then the composition of the k mechanisms $\mathcal{K}_{[k]}(D) = (\mathcal{K}_1(D), \mathcal{K}_2(D), ..., \mathcal{K}_k(D))$ is $\left(\sum_{i=1}^{k} \epsilon_i\right)$-differentially private.*

Similarly, under sequential composition, in (ϵ, δ)-DP mechanisms, the values for δ also sum up. The following theorem is stated in [66] without a proof. We include it here for the sake of completeness.

Theorem 3.7. *Let $\mathcal{K}_i$ with $\mathrm{Im}(\mathcal{K}_i) = S_i$ be an (ϵ_i, δ_i)-differentially private algorithm for $i \in [k]$, then the composition $\mathcal{K}_{[k]}(D_1) = (\mathcal{K}_1(D_1), \mathcal{K}_2(D_1), \ldots, \mathcal{K}_k(D_1))$ offers a level of $\left(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \delta_i\right)$-DP [66].*

3.1.6.2 Parallel Composition

While general sequences of queries accumulate privacy costs additively, when queries are applied to disjoint subsets of the data, the bound can be improved. When the input dataset is partitioned into disjoint subsets that are each subject to DP-analyses, then the total level of privacy does only depend on the worst privacy guarantee of all analyses, not on the sum [112].

Theorem 3.8 (DP for parallel composition). *Let there be n DP-mechanisms $\mathcal{K}_i$ with $\mathrm{Im}(\mathcal{K}_i) = S_i$ each providing ϵ_i-DP when being computed on disjoint subsets $D^{(i)}$ of the input domain D. Let $\mathfrak{K} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$. Furthermore, let $r_i = \mathcal{K}_i(D^{(i)})$ be their output. Then, any function $g(r_1, \ldots, r_n)$ with $g : 2^{\mathfrak{K}} \times \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ is $\max \epsilon_i$-DP[1].*

Proof. Let D_1 and D_2 be neighboring databases and let $D_1^{(i)}$ and $D_2^{(i)}$ be their i^{th} partition. As the databases are neighboring, they differ in at most one element. Let this element be in the j^{th} partition. Then, $D_1^{(j)} \neq D_2^{(j)}$ and $D_1^{(i)} = D_2^{(i)}$ for all $i \neq j$. Let $\mathcal{K}$ denote a sequence of $\mathcal{K}_i$ that are executed on the dataset D. Then, for any sequence s of outcomes $s_i \in \mathcal{K}_i(D^{(i)})$, the probability is $\Pr[\mathcal{K}(D) = s] =$

[1] The function can be anything that combines the outputs of the mechanisms, like the sum, the mean value, etc.

$\prod_i \Pr[\mathcal{K}_i(D^{(i)}) = s_i]$. Applying the definition of DP for each $\mathcal{K}_i$, we have

$$\begin{aligned}
\Pr[\mathcal{K}(D_1) = s] &= \prod_i \Pr[\mathcal{K}_i(D_1^{(i)}) = s_i] \\
&= \Pr[\mathcal{K}_j(D_1) = s_j] \cdot \prod_{i \neq j} \Pr[\mathcal{K}_i(D_1^{(i)}) = s_i] \\
&= \Pr[\mathcal{K}_j(D_1) = s_j] \cdot \prod_{i \neq j} \Pr[\mathcal{K}_i(D_2^{(i)}) = s_i] \\
&\leq e^{\epsilon_j} \Pr[\mathcal{K}_j(D_2) = s_j] \cdot \prod_{i \neq j} \Pr[\mathcal{K}_i(D_2^{(i)}) = s_i] \\
&\leq e^{\max_i \epsilon_i} \prod_i \Pr[\mathcal{K}_i(D_2^{(i)}) = s_i] \\
&\leq e^{\max_i \epsilon_i} \Pr[\mathcal{K}(D_2) = s]
\end{aligned}$$

The transition from line 4 to line 5 holds because $e^{\epsilon_j} \leq e^{\max_i \epsilon_i}$. □

3.1.6.3 Advanced Composition

There is also the concept of advanced composition that covers more complex scenarios. In addition to allowing the repeated use of DP mechanisms on the same database, it should also be possible to use different DP mechanisms on different databases that may nevertheless contain information relating to the same individual. This is a possible scenario, as new databases are created all the time. Adversaries may even influence the construction of these new databases, thus, in this setting, privacy is a fundamentally different problem than repeatedly querying a single, fixed database [66]. We call this setting k-fold adaptive composition.

Theorem 3.9 (DP for advanced composition, adapted from[66, Theorem 3.20])**.** *For all $\epsilon, \delta, \delta' \geq 0$, the class of (ϵ, δ)-DP mechanisms satisfies $(\epsilon', k\delta + \delta')$-DP under k-fold adaptive composition for*

$$\epsilon' = \epsilon\sqrt{2k\ln(1/\delta')} + k\epsilon(e^{\epsilon} - 1).$$

For the proof, see Theorem 3.20 in [66].

Note. Composition and group privacy are not the same thing and the improved composition bounds in Theorem 3.9 cannot yield the same gains for group privacy, even when $\delta = 0$.

3.2 Basic Mechanisms

After formalizing the notion of DP and its properties, the following section focuses on the basic mechanisms achieving this kind of privacy. We introduce those mechanisms because they represent the basic building blocks for any advanced mechanism, like the real-world applications depicted in Chapter 5.

3.2.1 Local and Global DP

In general, there are two different categories of DP mechanisms: *local* and *global*. According to [63], in a local approach, the data is perturbed at input time. There is no trusted data curator, thus, every person is responsible for adding noise to their own data before they share it. In global approaches, the data is perturbed at output time. For the global setup, every user needs to trust the data curator. Figure 3.4 visualizes both approaches.

The local approach is a more conservative and safer model. Under local privacy, each individual data point is extremely noisy, and thus, not very useful on its own. With large numbers of data points, however, the noise can be filtered out to perform meaningful analyses on the dataset. The global approach is, in general, more accurate as the analyses happen on "clean" data and only a small amount of noise needs to be added at the end of the process [38].

3.2.2 Randomized Response

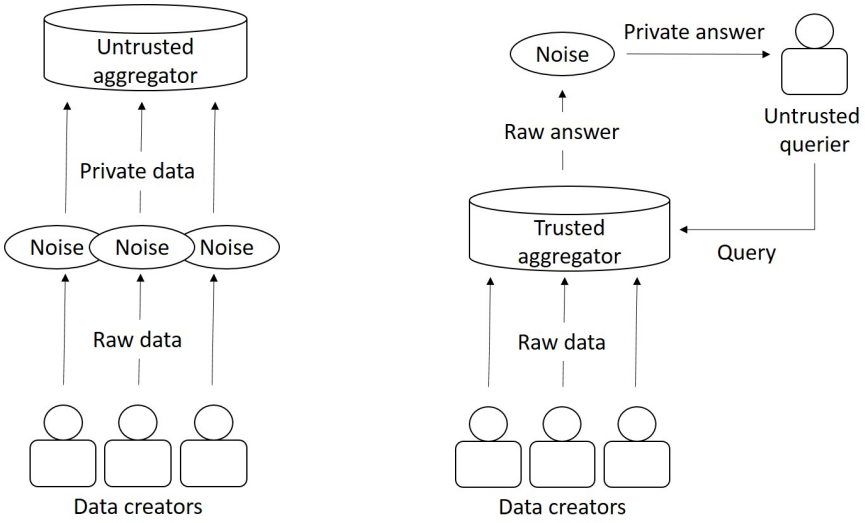
An early and very simple mechanism of local DP is *randomized response*, a technique developed in social science to collect statistical information about embarrassing or illegal behaviors [157].

Protocol

A simple yes-no question, that would otherwise give some private information away, can be answered by behaving according to the following protocol:

1. Toss a coin.

2. If tails, then respond truthfully.



(a) Local privacy.

(b) Global privacy.

Figure 3.4: In a local DP mechanism, each individual (data creator) adds noise to his real data before allowing it in a database for analysis purpose. This is necessary if the data curator is no trusted entity. In a global DP mechanism, there is a trusted curator who collects the users' data. When the user data is queried, the curator adds noise to the answers to achieve DP. Figures adapted from [38].

3. If heads, then toss a second coin and respond "Yes" if heads and "No" if tails.

The process is also visualized in Figure 3.5.

The privacy comes from the plausible deniability of any outcome. The "Yes" answer is not incriminating anymore, since it occurs with probability at least $\frac{1}{4}$, whether or not it is true for the participant. To find out the real percentage p of the "Yes" answers, we can use the fact, that with perfect randomization, the expected number of "Yes" answers is $\frac{1}{4}$ times the number of participants not having the property and $\frac{3}{4}$ times the number of participants having the property [66]. Thus, with p being the real percentage of a "Yes" answers, the observed percentage q of "Yes" answers is $q = \frac{1}{4}(1-p) + \frac{3}{4}p = \frac{1}{4} + \frac{p}{2}$. So, after the survey, one can calculate the expected percentage p from the observed percentage q by

$$p = 2\left(q - \frac{1}{4}\right). \tag{3.7}$$

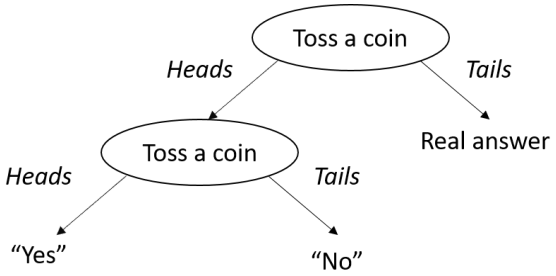


Figure 3.5: Visualization of the protocol of randomized response.

Accuracy

Of course, in reality $\Pr[\text{HEADS}] = \Pr[\text{TAILS}] = 0.5$ only holds, if we toss a coin n times and n converges towards infinity. One has to keep this is mind when trying to calculate the real number of "Yes" answers from the observed number. As stated in Section 3.2.1, in any local DP approach, each data point is very noisy and only with a large number of data points, the noise can be filtered out correctly.

To show how the accuracy of randomized responses depends on the number n of participants, we implemented a simulation of the process. Therefore, we created a datasets representing a survey that aims to find out how many smokers there are in a population. The dataset contains 3000 records, 600 of which represent smokers.

Note. In the following, we will show, that the accuracy of the mechanism depends only on the number n of participants and not on the percentage p of real "Yes" answers. Therefore, we could have chosen any value for the percentage of smokers.

We then implemented a subsampling that, given a size n', selects n' elements from the database according to the original distribution of smokers and non-smokers. We subsampled for all $n' \in \{5, 10, 15, ..., 3000\}$ to ensure that there are exactly 20 % of smokers in each subsample. On each of the samples, we executed the randomized responses protocol and counted the number of observed "Yes" answers to the question "Are you a smoker?". We then used Equation (3.7) to try to restore the real percentage of "Yes" answers. Figure 3.6 displays the estimated percentages and compares them to the real percentage. The diagram shows that with a growing number of participants, the accuracy of the estimated result increases. But especially in small datasets, the results obtained with randomized responses are barely useful to reflect the real data.

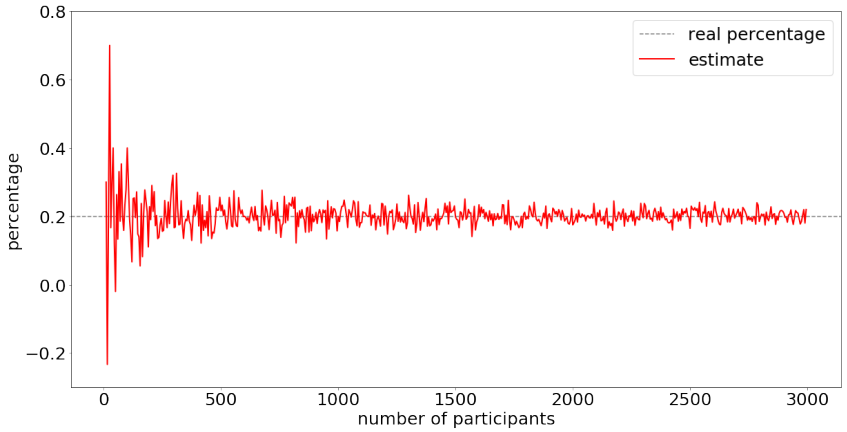


Figure 3.6: The x-axis represents the number of participants who have provided their answer through the randomized response protocol. On the y-axis, the percentage of smokers that could be calculated from the given answers is displayed. The real percentage of smokers in the dataset and every subsample is exactly 0.2.

Level of Privacy

Consider the previously described randomized response protocol executed with fair coin tosses:

Theorem 3.10 (adapted from [66, Claim 3.5])**.** *The randomized response technique yields* $\log 3$*-DP.*

Proof (adapted from [66]). It is possible to reason in a similar way for each answer, so we focus on the answer "Yes". A case analysis shows that $\Pr[Response = Yes|Truth = Yes] = \frac{3}{4}$. This is, because when the truth is "Yes", the outcome will be "Yes", if the coin comes up tails (probability $\frac{1}{2}$) or the first and second come up heads (probability $\frac{1}{4}$). Applying the same reasoning to the case of a "No" answer, we obtain

$$\frac{\Pr[Response = Yes|Truth = Yes]}{\Pr[Response = Yes|Truth = No]} = \frac{\frac{3}{4}}{\frac{1}{4}} = \frac{\Pr[Response = No|Truth = No]}{\Pr[Response = No|Truth = Yes]} = 3 = e^{\log(3)}. \tag{3.8}$$

□

Generalization of the Protocol

The idea of randomized responses can be generalized to probabilities α, β different from 0.5, where α is the probability that you have to answer truthfully and β the probability that you have to answer "Yes" in the second round [68]. Each answer of a participant i of the survey can be modeled with a Bernoulli random variable X_i which takes value 0 for "No" and 1 for "Yes". Let p be the real percentage of "Yes" answers over the whole population. We know that

$$Pr(X_i = 1) = \alpha p + (1 - \alpha)\beta.$$

Solving for the percentage of real "Yes" answers p yields

$$p = \frac{Pr(X_i = 1) - (1 - \alpha)\beta}{\alpha}.$$

When carrying out the protocol, due to the introduction of randomness, we cannot restore p, but only make an estimate $\hat{p}$. With a sample size of n, it is possible to estimate $Pr(X_i = 1)$ with $\frac{\sum_{i=1}^{n} X_i}{n}$. The estimate $\hat{p}$ of p is then

$$\hat{p} = \frac{\frac{\sum_{i=1}^{n} X_i}{n} - (1 - \alpha)\beta}{\alpha}.$$

To determine how accurate $\hat{p}$ is, we can compute its standard deviation σ through its variance. Assuming all individual responses X_i are independent and using the basic properties of variance, we end up with

$$\begin{aligned}
Var(\hat{p}) &= Var\left(\frac{\frac{\sum_{i=1}^{n} X_i}{n} - (1 - \alpha)\beta}{\alpha}\right) \\
&= Var\left(\frac{\frac{\sum_{i=1}^{n} X_i}{n}}{\alpha}\right) \\
&= Var\left(\frac{\sum_{i=1}^{n} X_i}{n\alpha}\right) \\
&= \frac{1}{n^2\alpha^2} Var\left(\sum_{i=1}^{n} X_i\right) \\
&= \frac{1}{n^2\alpha^2} Var(nX_i) \\
&= \frac{Var(X_i)}{n\alpha^2}.
\end{aligned}$$

The standard deviation σ is defined as the square root of the variance. Thus, σ is proportional to $\frac{1}{\sqrt{n}}$. Multiplying σ by the number of participants n yields the accuracy of the estimate. One can easily see that the accuracy is proportional to $\sqrt{n}$ (and does not depend on the real percentage p). Furthermore, Equation (3.8) in the proof of Theorem 3.10 on page 30 suggests that the privacy parameter ϵ can be tuned by varying α and β [128].

3.2.3 Laplace Mechanism

The Laplace mechanism is a global DP mechanism that is most often used to ensure privacy in numeric queries. Numeric queries or, mathematically, functions $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, are one of the most fundamental types of database queries. They map databases to k real numbers.

Notation

An important metric for the Laplace mechanism is the ℓ_1 sensitivity of a query function.

Definition 3.11 (ℓ_1 sensitivity, adapted from [66, Definition 3.1])**.** The ℓ_1 sensitivity $\triangle f$ of a function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ id determined over all pairs of neighboring databases $D_1, D_2 \in \mathbb{N}^{|\mathcal{X}|}$ as:

$$\triangle f = \max_{\substack{D_1, D_2 \\ \|D_1 - D_2\|_1 \leq 1}} (\|f(D_1) - f(D_2)\|_1).$$

This captures by how much a single individual's data can change the output of function f in the worst case [66]. In our example in Section 3.1.2, we worked with the numbers of smokers in a database. The ℓ_1 sensitivity of a count function (*counting query*) on that type of data is 1, because one individual can change the result of the count by at most one, depending on whether he is a smoker or not.

Intuitively, it makes sense that, if any individual can change the output of the query function a lot, we need to introduce more random noise to hide his participation. One possible distribution from which the random noise can be sampled is the Laplace distribution visualized in Figure 3.7. It is a symmetric version of the exponential distribution.

Definition 3.12 (Laplace distribution, adapted from [66, Definition 3.2])**.** The Laplace distribution (centered at 0) with scale b is the distribution with prob-

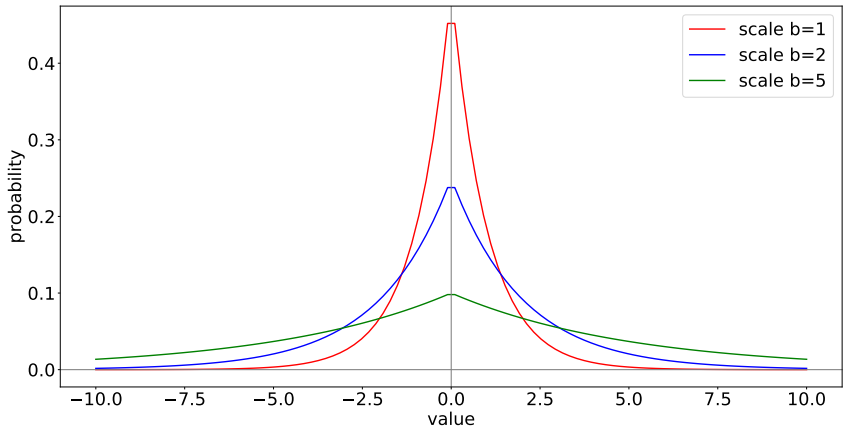


Figure 3.7: Different Laplace distributions for different scales b.

ability density function:

$$\text{Lap}(x, b) = \frac{1}{2b} e^{-\frac{|x|}{b}}.$$

The variance of this distribution is $\sigma^2 = 2b^2$. We use the notation $\text{Lap}(b)$ to denote a Laplace distribution with scale b.

Mechanism

The *Laplace mechanism*, which was first introduced by [63], works by computing a function f on the data and perturbing each output coordinate with noise drawn from the Laplace distribution. The scale of the noise needs to be calibrated to $b = \triangle f / \epsilon$ [2].

Definition 3.13 (Laplace mechanism, adapted from [66, Definition 3.3])**.** Given any function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, the Laplace mechanism is defined as:

$$\mathcal{K}_L(D, f, \epsilon) = f(D) + (Y_1, ..., Y_k)$$

where the Y_i are independent and identically distributed random variables drawn from $\text{Lap}(\triangle f / \epsilon)$.

Note. Instead of using noise from the Laplace distribution, one can also use Gaussian noise with variance calibrated to $\triangle f \log(1/\delta)/\epsilon$ to achieve (ϵ, δ)-DP. Both

[2] The division by ϵ influences the scale of the distribution as follows: if we divide by an ϵ with $0 < \epsilon < 1$, the scale gets larger, if we divide by an ϵ with $\epsilon \geq 1$, the scale gets smaller. If we recap the meaning of ϵ, this is what we want: smaller ϵ lead to the fact that larger amount of noise (from a distribution with larger scale) are added, which yields better privacy.

mechanisms behave similarly under composition, however, the use of the Laplace mechanism is better because it allows $\delta = 0$, which is not possible with Gaussian noise [66].

Level of Privacy

Theorem 3.14 (adapted from [66, Theorem 3.6])**.** *The Laplace mechanism provides ϵ-DP.*

Proof. Let $D_1, D_2 \in \mathbb{N}^{|\mathcal{X}|}$ two neighboring databases, and let f be some function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$. Let p_1 denote the probability density function of $\mathcal{K}_L(D_1, f, \epsilon)$ and p_2 denote the probability density function of $\mathcal{K}_L(D_2, f, \epsilon)$. We compare p_1 and p_2 at some arbitrary point $z \in \mathbb{R}^k$. By using the formula of the Laplace distribution from Definition 3.12, we obtain the following:

$$
\begin{aligned}
\frac{p_1(z)}{p_2(z)} &= \frac{\prod_{i=1}^{k} \left(\frac{1}{2\frac{\triangle f}{\epsilon}} \exp\left(- \frac{|f(D_1)_i - z_i|}{\frac{\triangle f}{\epsilon}} \right) \right)}{\prod_{i=1}^{k} \left(\frac{1}{2\frac{\triangle f}{\epsilon}} \exp\left(- \frac{|f(D_2)_i - z_i|}{\frac{\triangle f}{\epsilon}} \right) \right)} \\
&= \frac{\prod_{i=1}^{k} \left(\exp\left(- \frac{|f(D_1)_i - z_i|}{\frac{\triangle f}{\epsilon}} \right) \right)}{\prod_{i=1}^{k} \left(\exp\left(- \frac{|f(D_2)_i - z_i|}{\frac{\triangle f}{\epsilon}} \right) \right)} \\
&= \prod_{i=1}^{k} \frac{\exp\left(- \frac{|f(D_1)_i - z_i| \epsilon}{\triangle f} \right)}{\exp\left(- \frac{|f(D_2)_i - z_i| \epsilon}{\triangle f} \right)} \\
&= \prod_{i=1}^{k} \exp\left(\frac{\epsilon(|f(D_2)_i - z_i| - |f(D_1)_i - z_i|)}{\triangle f} \right) \\
&\leq \prod_{i=1}^{k} \exp\left(\frac{\epsilon(|f(D_1)_i - f(D_2)_i|)}{\triangle f} \right) \\
&= \exp\left(\frac{\epsilon(\|f(D_1) - f(D_2)\|_1)}{\triangle f} \right) \\
&\leq \exp(\epsilon).
\end{aligned}
$$

The first $\leq$ is due to the triangle inequality and the second one due to the definition of the sensitivity being the maximum over all ℓ_1 norms. As a consequence, $\triangle f \geq \|f(D_2) - f(D_1)\|_1$ for all possible D_1, D_2 and hence $\frac{(\|f(D_2) - f(D_1)\|_1)}{\triangle f} \leq 1$. □

Accuracy

To evaluate the accuracy of the results obtained with the Laplace mechanism, we implemented a simulation using the same dataset that we created to evaluate the accuracy of randomized responses. We performed the same type of subsampling as described above.

To facilitate the implementation, we then modeled the question of the percentage of smokers in a population by counting queries. According to Definition 3.13, DP in counting queries can be achieved by adding noise drawn from $\text{Lap}(1/\epsilon)$. This is because, as mentioned above, counting queries have sensitivity 1. In the simulation, we set the level of privacy to $\epsilon = \log 3$ to enable a comparability to the results obtained with randomized responses.

During the simulation, we counted the number of smokers in each of the subsamples, added random noise from $\text{Lap}(1/\log 3)$ and returned the result. At the end, we divided the result by the number n' of elements in the sample to determine the percentages. Figure 3.8 displays the noisy estimates and the real percentage of smokers in the dataset. We see that the expected distortion is independent of the size of the database[3]. The level of noise in the data is, thus, only dependent on the choice of ϵ (and the sensitivity of the data).

3.2.4 Exponential Mechanism

The third basic mechanism is the *exponential mechanism* that was introduced by McSherry and Talwar [113]. It is, as well as the Laplace mechanism, a global DP mechanism and it can operate on use cases where the two previously introduced mechanisms fail.

Use Cases

The Laplace mechanism works well on data whose usefulness is relatively unaffected by additive perturbations, e.g., in counting queries. However, there are cases where the addition of noise leads to a useless result. Take as an example a digital

[3]For very small sample sizes, the estimates exhibit a certain amount of noise. This can be explained by looking at $\text{Lap}(1/\log 3) \approx \text{Lap}(1)$ which is visualized in Figure 3.7 on page 33. Due to the variance of the distribution $2b^2 = 2$, noise values up to about 2 or -2 are not unlikely to appear. In a subsample with size 5, there is only 1 smoker, thus, if noise with the amount 2 is added, the distortion is very high. However, in comparison to randomized responses, the estimated value of "Yes" answers converges to the real value already for quite small sample sizes.

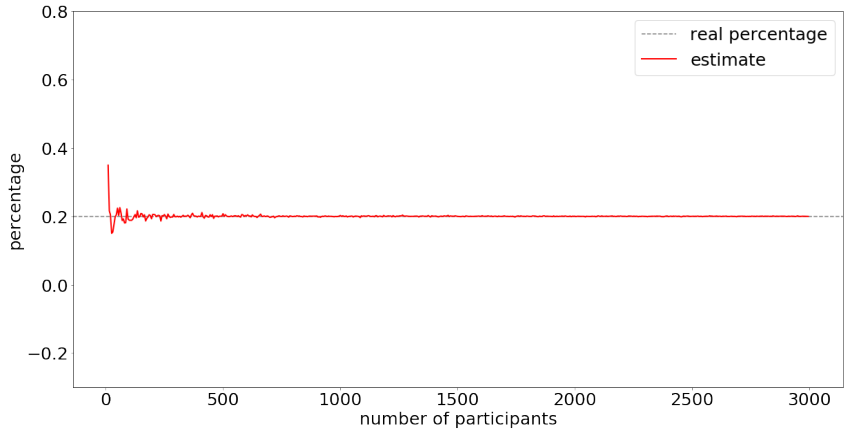


Figure 3.8: Simulation of the Laplace mechanism. The x-axis depicts the size of the subset that the mechanism was executed on, the y-axis shows the calculated percentage of smokers from the noisy responses. The real percentage of smokers in every subset is exactly 0.2. The scale of the y-axis in this diagram is the same as in Figure 3.8 to make comparisons more intuitive.

goods auction. In such an auction, a seller has an unlimited supply of certain items he wants to sell, like digital movies. For each movie, he wants to set the optimal price that maximizes his profit. Given the situation that there are four potential buyers. The first three are willing to spent 1 € on the movie, the fourth one is willing to spend 4.01 €. By setting the price to 1 € the profit is 4 €, with a selling price of 4.01 €, the profit is maximized to 4.01 €; but if the vendor sets the price to 4.02 € his profit is 0 €. This shows that the function to find the optimal fixed price to sell the items is not robust to additive noise. Neither is it insensitive, as a single bidder has the potential to shift the optimal price arbitrarily.

Another situation that cannot be captured by randomized responses or the Laplace mechanism appears in analyses with non-numeric output. E.g., if we want to learn a certain property about a population, like "What is the most common eye color in Germany?", or in machine learning, where a classifier can predict several non-numeric categories.

Mechanism

Assume we are given some arbitrary range $\mathcal{R}$. For our examples with the digital goods auction or the eye colors, the ranges would be

- $\mathcal{R} = \{1.00, 1.01, 1.02, \ldots\}$ and
- $\mathcal{R} = \{\text{Blue, Green, Brown, Grey}\}$.

For the exponential mechanism, we need some utility function $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \to \mathbb{R}$, which maps pairs of database and output elements to utility scores. $u(D, r)$ represents how "good" the output r is for the database D. In the digital auction example, the utility for a certain price would simply be the profit obtained with this price on the given database of users. The sensitivity of u, $\triangle u$ is measured with respect to its database arguments. It can be arbitrarily sensitive in its range argument.

Definition 3.15. The sensitivity of the utility function $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \to \mathbb{R}$ is given by

$$\triangle u = \max_{r \in \mathcal{R}} \max_{\substack{D_1, D_2 \\ \|D_1 - D_2\|_1 \leq 1}} |u(D_1, r) - u(D_2, r)|.$$

The idea of the exponential mechanism is to answer the posed queries with elements from the real data. Each element is returned with a certain probability. Outputs with high utility should be exponentially more likely than outputs with low utility. E.g., prices in the digital goods auction that cause a high profit should be returned with an exponentially higher probability than prices that cause a low profit. Intuitively, the rate of likeliness depends on the sensitivity of the utility function and the privacy budget ϵ. For a database D,each possible $r \in \mathcal{R}$ is returned with a probability proportional to $\exp\left(\frac{\epsilon u(D,r)}{\triangle u}\right)$ and so the privacy loss can be approximated by

$$\log\left(\frac{\exp\left(\frac{\epsilon u(D_1,r)}{\triangle u}\right)}{\exp\left(\frac{\epsilon u(D_2,r)}{\triangle u}\right)}\right) = \frac{\epsilon(u(D_1, r) - u(D_2, r))}{\triangle u} \leq \epsilon.$$

New data records that are added into the database could cause the utility of one element to decrease and others to increase. If we, for example, add a user who is willing to pay 3€ to the digital goods database, the maximum profit would not be obtained by setting the price to 4.01€, but by setting it to 3€. The utility of the price 4.01€ would thereby not be the optimal one anymore, whereas the utility of 3€ would raise from 3€ to 6€, the highest profit possible. The actual mechanism reserves half the privacy budget for this kind of changes, resulting in the following formulation:

Definition 3.16 (Exponential mechanism, adapted from [66, Definition 3.4])**.** The exponential mechanism $\mathcal{K}_E$ on database D outputs an element $r \in \mathcal{R}$ with probability proportional to $\exp\left(\frac{\epsilon u(D,r)}{2\triangle u}\right)$.

Theorem 3.17 (adapted from [66, Theorem 3.10]). *The exponential mechanism provides ϵ-DP.*

Proof. We consider the probabilities that the exponential mechanism outputs a certain element $r^* \in \mathcal{R}$ on the two neighboring databases $D_1, D_2 \in \mathbb{N}^{|\mathcal{X}|}$. We call the element in which the databases differ d_1 and d_2 respectively.

$$
\begin{aligned}
\frac{\Pr[\mathcal{K}_E(D_1, u, \mathcal{R}) = r^*]}{\Pr[\mathcal{K}_E(D_2, u, \mathcal{R}) = r^*]} &= \left(\frac{\frac{\exp\left(\frac{\epsilon u(D_1, r^*)}{2\triangle u}\right)}{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_1, r)}{2\triangle u}\right)}}{\frac{\exp\left(\frac{\epsilon u(D_2, r^*)}{2\triangle u}\right)}{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_2, r)}{2\triangle u}\right)}} \right) \\
&= \left(\frac{\exp\left(\frac{\epsilon u(D_1, r^*)}{2\triangle u}\right)}{\exp\left(\frac{\epsilon u(D_2, r^*)}{2\triangle u}\right)} \right) \left(\frac{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_2, r)}{2\triangle u}\right)}{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_1, r)}{2\triangle u}\right)} \right) \\
&= \exp\left(\frac{\epsilon(u(D_1, r^*) - u(D_2, r^*))}{2 \triangle u} \right) \left(\frac{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_2, r)}{2\triangle u}\right)}{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_1, r)}{2\triangle u}\right)} \right) \\
&\leq \exp\left(\frac{\epsilon}{2}\right) \left(\frac{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_2, r)}{2\triangle u}\right)}{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_1, r)}{2\triangle u}\right)} \right) \\
&= \exp\left(\frac{\epsilon}{2}\right) \left(\frac{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_2 \setminus \{d_2\}, r) + \epsilon u(\{d_2\}, r)}{2\triangle u}\right)}{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_1 \setminus \{d_1\}, r) + \epsilon u(\{d_1\}, r)}{2\triangle u}\right)} \right) \\
&= \exp\left(\frac{\epsilon}{2}\right) \left(\frac{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_2 \setminus \{d_2\}, r)}{2\triangle u}\right) \sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(\{d_2\}, r)}{2\triangle u}\right)}{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(D_1 \setminus \{d_1\}, r)}{2\triangle u}\right) \sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(\{d_1\}, r)}{2\triangle u}\right)} \right) && (3.9) \\
&= \exp\left(\frac{\epsilon}{2}\right) \left(\frac{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(\{d_2\}, r)}{2\triangle u}\right)}{\sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon u(\{d_1\}, r)}{2\triangle u}\right)} \right) && (3.10) \\
&= \exp\left(\frac{\epsilon}{2}\right) \sum_{r \in \mathcal{R}} \exp\left(\frac{\epsilon(u(\{d_2\}, r) - u(\{d_1\}, r))}{2 \triangle u} \right) && (3.11) \\
&= \exp\left(\frac{\epsilon}{2}\right) \exp\left(\frac{\epsilon(u(\{d_2\}, d_2) - u(\{d_1\}, d_1))}{2 \triangle u} \right) && (3.12) \\
&\leq \exp\left(\frac{\epsilon}{2}\right) \exp\left(\frac{\epsilon}{2}\right) = \exp(\epsilon).
\end{aligned}
$$

The first $\leq$ results from the estimation of $\frac{|u(D_1,r^*)-u(D_1,r^*)|}{\triangle u} \leq 1$ because of the definition of the sensitivity of the utility function as the maximum over all pairs D_i, D_js and all $r \in \mathcal{R}$. To get from (3.9) to (3.10), we use the fact that $D_1 \setminus \{d_1\} = D_2 \setminus \{d_2\}$. From (3.11) to (3.12), we use the fact, that from a dataset that contains only one element, we can only output this one element. Lastly, for the second $\leq$, we use the definition of the sensitivity of the utility function again. □

3.3 Comparison of mechanisms

Now, that we have introduced the three basic mechanisms for DP, we can compare their qualities. An overview is provided in Table 3.1. We see that the Laplace and the exponential mechanism can achieve any level ϵ of privacy, whereas the amount of privacy in the original protocol of randomized responses with $\alpha = \beta = 0.5$ is fixed with $\log 3$. The accuracy of randomized responses increases with the number of data points used. For the other mechanisms, the accuracy of the output depends on the chosen privacy parameter ϵ and the sensitivity of the query or the utility function respectively. The use cases of randomized responses can also be covered by the other mechanisms, however, randomized responses offer an important advantage: no trusted data curator is needed. Thus, with sensitive data, the data generators might prefer this mechanism over the others because their real data is not saved by anyone.

	RR (coin toss)	Laplace	Exponential
Privacy	$(log(3), 0)$	$(\epsilon, 0)$	$(\epsilon, 0)$
Accuracy	Depends on n	Depends on ϵ and sensitivity of query function	Depends on ϵ and sensitivity of utility function
Application	Binary questions	Numerical queries	Non-numerical queries/ Data not robust to noise
Setting	Local	Global	Global

Table 3.1: Comparison of the three basic mechanisms of DP.

4 Implementation of a linear regression preserving DP

After having introduced the basic DP mechanisms, in this chapter, we will use them to implement a privacy-preserving linear regression. This experimental part of the thesis has two main goals. The first one is to determine the trade-off between good privacy and accurate machine learning results experimentally. Namely, we want to investigate how the choice of the privacy parameter ϵ influences the prediction quality of the linear regression. The second goal is to examine the practicability of DP in the context of machine learning. We chose linear regression because it is the most basic machine learning algorithm and widely known. Furthermore, it is a linear model, thus, its parameters and predictions are reproducible.

The following section gives a brief introduction to linear regression and how DP can be applied to it. Then, the dataset that was used for the experiments is presented. Afterwards, the two methods that were implemented to render linear regression differentially private are depicted. At the end of the chapter, the results of both methods are shown.

4.1 Linear Regression

Before applying DP to linear regression, we first give an introduction on the method itself.

4.1.1 Definition of Linear Regression

Linear regression is a supervised machine learning method. Its goal is to predict the value of one or more continuous target variables y, given the value of a D-dimensional vector $\vec{x}$ of input variables. More precisely, given a dataset with N records $\{\vec{x}_n\}, n \in [N]$ together with the corresponding target values y_n, the goal is to train a *model* that is able to predict the value of y for a new value of $\vec{x}$ [14].

In linear regression, the model is a function that represents the target value as a linear combination of the input variables

$$y = f(\vec{x}, \vec{\omega}) = \omega_0 + \omega_1 x_1 + \ldots + \omega_D x_D$$

where $\vec{x} = (x_1, \ldots, x_D)^T$ [14]. The values ω_i are often called weights.

The equation can also be written in matrix form. Let $\vec{y}$ denote the $(N \times 1)$-dimensional vector that contains the targets of the dataset with N records. Furthermore let X be the $(N \times D)$-dimensional matrix that contains the input variables for all data records. To account for ω_0, which is not dependent on any input variable, we construct $\bar{X}$ from X by appending a column to it that contains only ones. $\bar{X}$, therefore, has dimension $(N \times D + 1)$. Let $\vec{\omega}$ be the $(D + 1 \times 1)$-dimensional weight vector. Then

$$\vec{y} = \bar{X}\vec{\omega}. \tag{4.1}$$

In machine learning, the process of building the model from a dataset is often referred to as *training*. The dataset that is used for the training is, therefore, called the *training dataset*. In linear regression, the training process aims to find a weight vector $\vec{\omega}$ that is the best approximation for Equation (4.1). The quality of an approximation can be measured by a so-called *cost function*. The cost function measures the distance between the predictions of the model $\vec{y}^*$ and the true targets $\vec{y}$. In linear regression, a common cost function is the *mean squared error*. It is defined by

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - y_i^*)^2.$$

After the weights are determined by the training process, for a new element $\vec{x}$, the target y^* can be predicted by multiplying $\vec{x}$ with the weights. The squared error $(y - y^*)^2$ or the error $\sqrt{(y - y^*)^2}$ indicate the quality of the prediction.

4.1.2 DP in Linear Regression

DP relies on the addition of noise to dissimulate the participation of an individual in the dataset. In linear regression, there are several points where noise can be added, some more suitable than others.

The Dataset. Independently of the machine learning method that we want to apply, we can privatize the dataset itself by adding noise to the data. There are several publications about differentially private data publication, e.g. [26, 34, 162]. The main idea behind it is to learn the distribution of the data and to perform a differentially private synthetic data generation based on this distribution. All analyses can then be conducted on the synthetic data instead of the sensitive one. [96] proposed such synthetic data generation for linear regression. His method first uses the Laplace mechanism to produce a noisy multi-dimensional histogram of the original data. Then, it generates a synthetic dataset according to the statistics in the noisy histogram. Finally, it uses this synthetic data to compute the regression model. This method offers the advantage that the amount of noise added to the data can be adapted dependent on the analyses that one wants to perform on it. It furthermore allows to execute the linear regression in a normal way, thus, separating the privacy and the analysis part. However, the method only works well for datasets with small dimensionality because of the complexity of the noisy histograms.

The Cost Function. Another point in linear regression, where noise can be added is the cost function. [23, 24] proposed solving a perturbed optimization problem to create a differentially private logistic regression for cost functions that are convex and doubly differentiable. [166] advanced a similar idea for linear regression with arbitrary cost functions and called it the *functional mechanism*. The approach will be explained more in detail in Section 4.3.2.

The Prediction Output. The last point in linear regression where noise can be added is the prediction outcome itself. [139] proposed a general framework for statistical analysis that utilizes both the Laplace mechanism and the exponential mechanism and performs the perturbation on the prediction output. The drawback of this method is that it requires the output space of the analyses to be bounded, otherwise, calculating the sensitivity for the noise is impossible. However, in linear regression, the output space is not bounded, thus, noise addition to the prediction output is only possible after making several assumptions. An example of how these assumptions would have to be made is given in Section 4.3.1.

4.2 The Boston Housing Dataset

We chose the Boston Housing dataset [82] for evaluation of the differentially private analyses because it is one of the standard datasets used for linear regression analysis. It consists of 506 data records, each containing 13 variables, so-called

features that can be used to predict the target variable, namely the housing *price* in $1000.

The features are [17]

1. *crim:* per capita crime rate per town,
2. *zn:* proportion of residential land zoned for lots over 25 000 sq.ft.,
3. *indus:* proportion of non-retail business acres per town,
4. *chas:* binary variable, indicating whether or not area bounds the Charles river,
5. *nox:* nitrogen oxides concentration (parts per 10 million),
6. *rm:* average number of rooms per dwelling,
7. *age:* proportion of owner-occupied units built prior to 1940,
8. *dis:* weighted mean of distances to five Boston employment centers,
9. *rad:* index of accessibility to radial highways,
10. *tax:* full-value property-tax rate per $10 000,
11. *ptratio:* pupil-teacher ratio by town,
12. *black:* $1000(\mathrm{Bk}-0.63)^2$ where Bk is the proportion of blacks by town,
13. *lstat:* percentage of lower status of the population.

According to [66], correlation in data degrades the privacy guarantees. It is, therefore, useful to analyze the correlation between the different features in the dataset and to keep only a subset of features that do not exhibit strong correlation between each other but produce similarly good prediction results as the entire set when being used to train a linear regression. The procedure of finding such a subset of features is called *feature selection* [79]. [140] suggest that feature selection should be performed in a differential private manner as well because otherwise, the privacy of the individuals in the dataset can be breached. They proposed a DP feature selection mechanism. We will, however, for the sake of simplicity, use a conventional feature selection mechanism and treat the resulting features as our initial dataset.

Figure 4.1 depicts all 13 features from the Boston Housing dataset and their correlations over the data records. A value close to one indicates strong correlation,

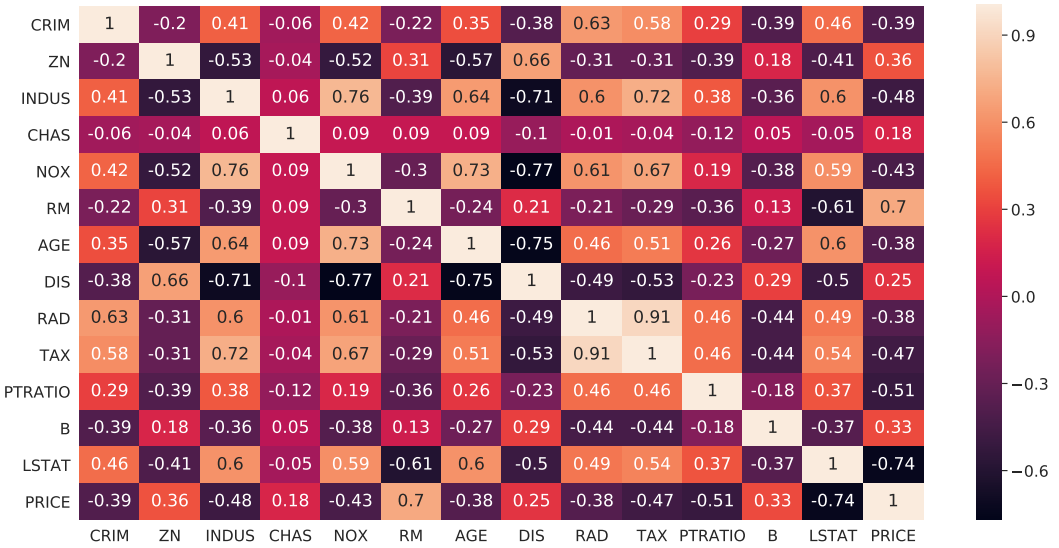


Figure 4.1: Heatmap displaying the correlations between the features and target variables in the Boston Housing dataset. Values close to one indicate strong correlation, values close to minus one indicate strong negative correlation.

a value close to minus one indicates strong negative correlation. The visualization indicates that the features that correlate most with the target *price* are *lstat* with a value of -0.74 and *RM* with 0.7. This suggests, that these features are well suited to be used when predicting the housing price. However, the correlation between these two features is quite high with a value of -0.61 and should, therefore, not both be in the selected feature subset.

To perform a feature selection that is not only based on the correlation between features and target we used the Python scikit-learn library [121]. This library offers a functionality called `SelectFromModel` that selects features based on the importance of the weights in the trained model. Given the Boston Housing dataset and a linear regression model, it selected features *chas*, *nox*, and *rm*.

To evaluate the performance of this feature subset we trained two linear regression models, one on the three features and one on all features and compared the resulting mean errors. We evaluated the mean errors over a 10-fold *cross-validation* [14, p. 33]. This means that we divided both datasets into 10 different partitions, each containing approximately 51 data records. We then trained and evaluated the model 10 times. Every time, the training was conducted on 9 of the partitions and the prediction was performed for the remaining partition. The mean error was then calculated as a mean over all 10 errors from the cross-validation. On the full dataset

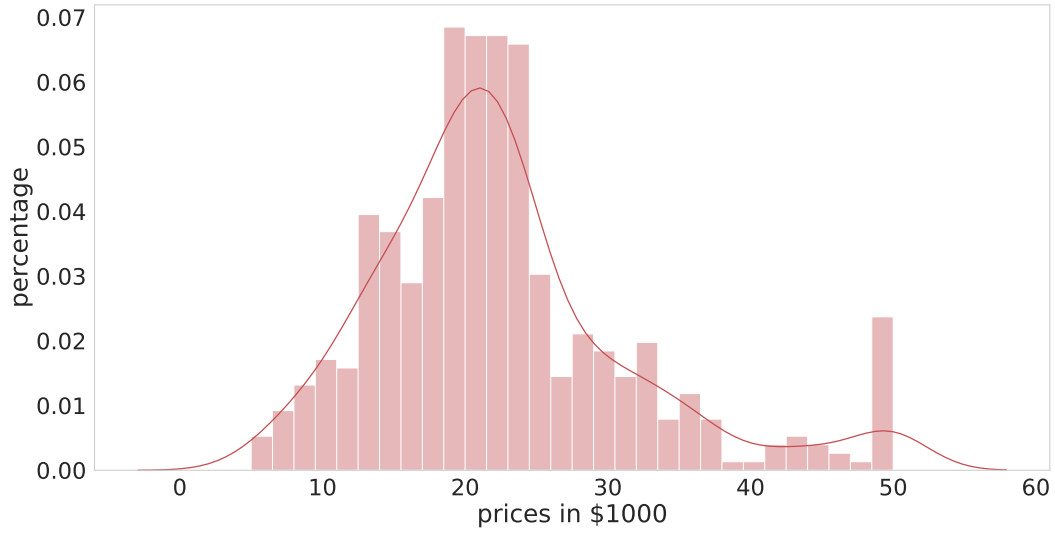


Figure 4.2: Distribution of housing prices in the Boston Housing dataset.

with all 13 features, the mean error was approximately 5.1808, and for the smaller dataset with the selected 3 features 5.9864.

When looking at the distribution of housing prices in the dataset depicted in Figure 4.2, we see that the price range is between 5 and 50 thousand dollars. A mean error of 5.1808, therefore, seems high. This indicates that the linear model is probably not best suited for price prediction on this data. But given this baseline, a mean error of 5.9864 on the three features seems tolerable.

Note. The distribution of the prices resembles a normal distribution with few outliers in the middle. However, there is a large peak at the maximum value, 50 thousand dollars. Even though not documented anywhere, this suggests that the prices were capped at $50 000.

4.3 Implementation

Based on the new dataset containing 506 data records with 3 features and one target variable each, we implemented two differentially private linear regressions, both based on the Laplace mechanism. The first implementation is a naive output perturbation method we developed. It adds noise drawn from the Laplace distribution to the regression's predicted values and serves as a baseline for comparison with the second method which is an implementation of the functional mechanism described in [166]. In both cases, the Laplace mechanism is most suited, because un-

like randomized responses, it is applied to data output and, unlike the exponential mechanism, it is applied to numerical data.

4.3.1 Method 1: Laplace Noise Addition to Prediction Output

For an implementation of the Laplace mechanism, it is necessary to determine the ℓ_1 sensitivity (see Definition 3.11 on page 32) of the function that is executed on the dataset. Without the sensitivity, it is impossible to pick the right scale of noise. The function, in this case, is the linear regression. Even though linear regression is a very basic type of analysis, its sensitivity is very hard to determine because it would involve analyzing how much the regression output would change if an arbitrary record in the dataset was modified. This is difficult because of the complex correlation between regression inputs and outputs [166].

Nevertheless, it is still possible to approximate the results for a specific dataset. This is what we did for the Boston Housing. To determine the sensitivity of the linear regression, we would have to calculate the ℓ_1 distances over all pairs of neighboring datasets and choose the maximum. This is, however, infeasible as there exist infinitely many datasets and, unlike with, e.g., counting queries, where the ℓ_1 distance for every two neighboring datasets is either 1 or 0, the ℓ_1 distances for a linear regression can possibly take any value.

As a first assumption, we, thus, fixed the Boston Housing dataset with its 506 data records as the first database, calculated the ℓ_1 distances to all of its possible neighboring databases, and took the maximum of the resulting values as the sensitivity. Yet, this approach has another drawback. There are still infinitely many possible neighboring databases. Furthermore, we could construct an element that would cause a maximum change in the regression output: Such an element would consist of values of positive or negative infinity, according to the corresponding weights. However, this element is no realistic database element. This leads to the question of defining a realistic database element: setting value ranges for every feature and for the target and deciding which value combinations are realistic and which ones are not. At this point, we would have to make too many assumptions that might, at least without the domain knowledge, be arbitrary and cause the output to be unrealistic.

Instead, we made another more realistic, second assumption. It is difficult to construct a neighboring database by changing one element in a way that causes large changes in the outcome while the new database still remains realistic. We, therefore, constructed a neighboring database by deleting the element that has the largest influence on the regression output from the first database. According to

[88], the definition of neighboring databases works equally on two databases with one element that is changed between them, and with one element that is deleted.

Intuitively, the element with the highest influence on the regression output is the one that, after training the model on all data elements, has the largest squared error. This means that it is situated furthest away from the multi-dimensional regression plane. In the Boston Housing dataset, this would be the data record at index 368. To confirm this intuition, we performed a brute force analysis over the entire dataset. With only 506 data records, this is computationally possible. Knowing the original mean squared error over the entire dataset, we trained 506 different linear regression models, each with one element from the dataset left out for training. For each of the models, we determined the mean squared error and saved the difference between it and the original mean squared error. The model with the largest mean square error difference (approximately 3.01) was the one where the element at index 368 was left out, thus confirming our intuition.

Given the element with the highest influence on the regression output, we then had the two neighboring databases to our disposal whose outputs differ most: the original Boston Housing dataset, and the dataset where the record at index 368 was deleted. The sensitivity is then defined as the difference between their outputs. We measured the difference between the two models by their prediction quality, i.e., by their mean squared error. As stated above, the mean square error difference between the two datasets was approximately 3.01 making it the value for our sensitivity.

With the sensitivity for this dataset approximated, we implemented an approximation of a differentially private linear regression. To do this, we trained the regression model on the original data and added noise drawn from $\mathrm{Lap}(3.01/\epsilon)$ to every prediction output for different values of ϵ. To select the value range for ϵ, we oriented ourselves by the values being discussed by different parties: In the literature, there is no consensus about what values of ϵ are "private enough". Most experts agree that values between 0 and 1 are very good, that values from 1 to 10 are various degrees of "better than nothing" and values above 10 do not imply good DP [38]. According to Dwork, the selection of ϵ is a social question and she tends to think of it as 0.01, 0.1 or, in some cases $\log 2$ or $\log 3$ [56, 57].

Figure 4.3 shows the results obtained by method 1. For each value of ϵ that we sampled from $\{0.01, 0.02, 0.03, \ldots, 10.0\}$, we determined the mean error by a 10-fold cross validation. The results depicted in the figure are the mean results over 100 repetitions of the experiment. The above-mentioned common values of ϵ are marked in the curve.

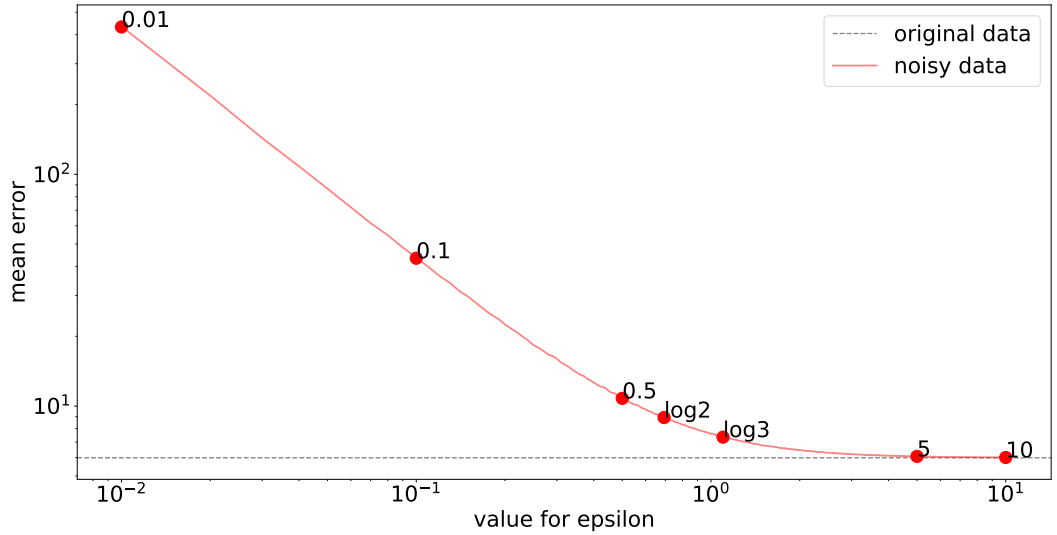


Figure 4.3: Results of method 1: prediction output perturbation by the Laplace mechanism. The x-axis depicts the values for ϵ the the y-axis depicts the corresponding mean error as an average over 100 repetitions, each with a 10-fold cross validation. Both axes are scaled logarithmically.

4.3.2 Method 2: Functional Mechanism

The functional mechanism [166] enforces ϵ-DP by perturbing the cost function of the linear regression. For this mechanism, all values for features and also the target variable need to be in the range $[-1, 1]$. We therefore had to perform a preprocessing on the Boston Housing dataset that implements a feature-wise normalization of the values to fit the range $[-1, 1]$. This was achieved by calculating the column-wise minimum and maximum value in the dataset, subtracting the corresponding minimums from all values and dividing every value by the column-wise scale that is the column-wise maximum minus the column-wise minimum.

We then implemented the mechanism as described in the paper and implemented by the authors [75, 166]. The idea behind the algorithm relies on the fact that not only the results of the cost function, but the function itself can be perturbed. Recall that training a regression model leads to a weight vector $\vec{\omega}$. Publishing this vector would violate privacy since it reveals information about the data. Addressing this issue by adding noise from the Laplace distribution directly to the components of $\vec{\omega}$ is challenging because it would involve a sensitivity analysis of $\vec{\omega}$ which is challenging due to the complex relation between the input values and the weights. However, it is possible to inject noise directly to the cost function instead.

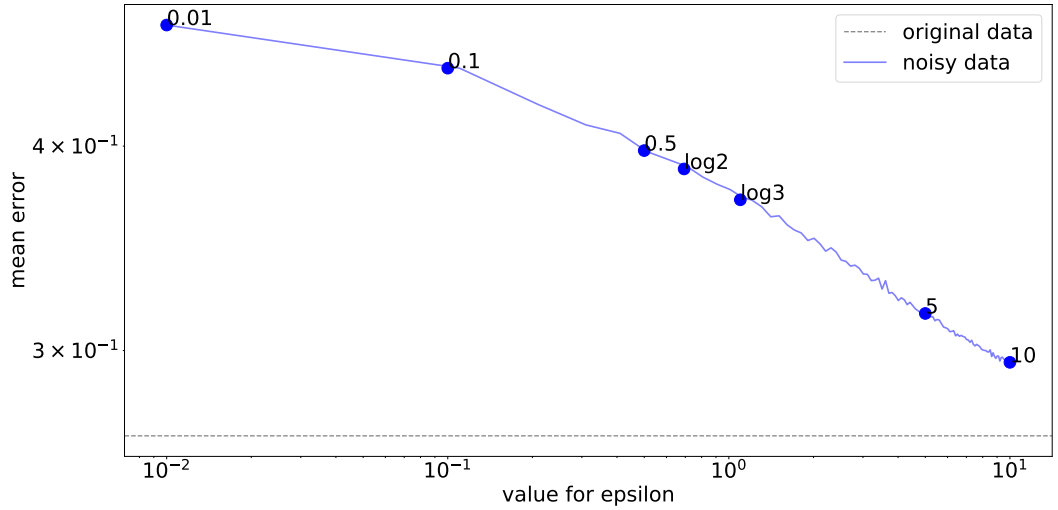


Figure 4.4: Results of method 2: objective function perturbation by the Laplace mechanism according to the functional mechanism [166]. The x-axis depicts the values for ϵ, the y-axis depicts the corresponding mean error as an average over 100 repetitions, each with a 10-fold cross validation. Both axes are scaled logarithmically.

To do so, the authors proposed writing the function as a potentially infinite polynomial of the weights and adding noise to the coefficients of the polynomial. The sensitivity that determines the amount of noise is calculated from the ℓ_1 distance of these coefficients over two neighboring databases. At the end, the weights that minimize the perturbed cost function are determined and used to predict new values.

Figure 4.4 shows the results obtained by this method. As the values for the features and the target variable are scaled to the range of $[-1, 1]$, the scale for the mean error is different from the original one. For each value of ϵ that we sampled from $\{0.01, 0.02, 0.03, \ldots, 10.0\}$, we determined the mean error by a 10-fold cross validation. The results depicted in the figure are the mean results over 100 repetitions of the experiment.

4.4 Results

Figure 4.3 and Figure 4.4 give a good impression on the results of the respective mechanisms. For better comparability, we scaled the mean errors of method 2 up by multiplying them with a factor that scales the original mean error of method 2 to the same value as in method 1. The results are depicted in Figure 4.5

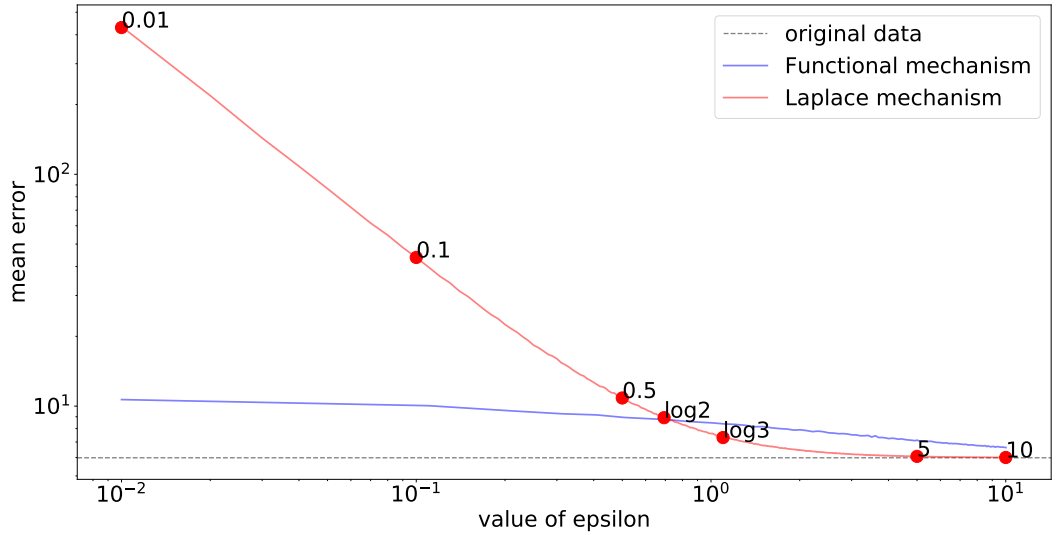


Figure 4.5: Results of method 1 and method 2 scaled to the same mean squared error value for better comparability. The x-axis depicts the values for ϵ, the y-axis depicts the corresponding mean error as an average over 100 repetitions, each with a 10-fold cross validation. Both axes are scaled logarithmically.

Given these results, it is possible to evaluate the model accuracy against different privacy parameters.

One can see that for the smallest value of ϵ that we sampled, 0.01, the mean error of method 1 is nearly 100 times as high as the original mean error. With ϵ increasing, the mean error in this method drops exponentially, until it reaches the original mean error at approximately $\epsilon = 5$. This indicated that for small ϵ (approximately $\epsilon < 0.5$), the prediction results returned by this method are useless. When examining the distribution of the housing prices again (Figure 4.2 on page 46) and remembering the shape of the Laplace distribution, we understand why the results for small ϵ are that poor. The distribution of the housing prices resembles a normal distribution around \$21 000. For such distributions, noise drawn from the Laplace mechanism is very suitable. However, due to the peak at \$50 000, the scale of noise drawn from the Laplace mechanism has to be chosen much larger to dissimulate the presence of data records with these large target values in the dataset. The resulting scale is too large for the housing prices in the normal distribution, rendering the average results highly inaccurate.

In method 2, for $\epsilon = 0.01$, the mean error of the noisy prediction is only approximately twice as large as the original mean error. With ϵ increasing, the mean error improves linearly with a small negative slope. Due to the perturbation of the cost

function, method 2 does not reach a mean error as low as the original one. However, for the values $\epsilon \leq \log 2$, it yields more precise results than method 1. As stated above, according to [56, 57], these are the values that should be chosen for ϵ. One also needs to keep in mind that the parameter ϵ is exponential. This implies that a system with $\epsilon = 1$ is almost 3 times more private than a system with $\epsilon = 2$ and over 8000 times more private than $\epsilon = 10$ [38].

5 Real-World Implementations of DP

In recent years, several application frameworks for DP have been proposed by different parties. Table 5.1 depicts an overview of the most popular ones and gives a brief summary of their respective purpose. In the following, the two approaches developed by the "big players", namely Google and Apple, are presented and their application in real-world data analysis is investigated.

5.1 Google's RAPPOR

In Chapter 1, we pointed out that software development enjoys a large benefit from analyzing user data. In 2014, Google proposed the *RAPPOR* (Randomized Aggregatable Privacy-Preserving Ordinal Response) algorithm to achieve this goal by retrieving statistics from end-user client software while preserving the users' privacy [69]. In the following sections, we depict the RAPPOR algorithm, its underlying data structures, the procedure to perform analyses on the privatized data, and the application of the algorithm in Google's own Chrome web browser implementation. At last, we present the limitations and possible drawbacks of the algorithm.

5.1.1 Algorithm

The RAPPOR algorithm relies heavily on randomized responses (see Section 3.2.2). The inventors chose this mechanism to provide local privacy, a setting in which no trusted data curator is needed. In this setting, users are most willing to contribute their data. However, the original protocol of randomized responses is not suited for scenarios with more than two possible answers. It can, furthermore, not represent settings in which a single user can provide multiple responses. Thus, the protocol had to be adapted for the RAPPOR algorithm.

To account for numerous possible answers and multiple answers per user, a bit array can be used. Each bit b_i in this bit array stands for a possible answer to the

Year	Framework	Purpose
2009	PINQ [112]	Privacy INtegrated Queries platform which provides a DP programming interface through an SQL-like language.
2010	Airavat model [130]	Integration of decentralized flow control and DP to provide strong security and privacy guarantees for MapReduce computations.
2012	GUPT [117]	DP framework that extends the idea of PINQ and Airavat. Also applicable to programs that were not developed with privacy in mind.
2014	Google's RAPPOR [69, 124]	Technology for crowdsourcing statistics from end-user client software with DP guarantees.
2015	DP in Telco Big Data Platform [84]	Implementation of three basic DP architectures in the Telco Big Data Platform for data mining.
2015	E-health data release under DP [97]	Implementation of an efficient e-health data release scheme with consistency guarantee under DP.
2016	Apple's iOS 10 [6]	System that enables identifying popular emojis, popular health data types, and media playback preferences in Safari over large groups of users with DP guarantees.
2018	Uber's SQL DP [41, 89]	Empirical study to determine the requirements for practical DP on a dataset of queries written by Uber employees. Practical methods to meet the requirements of DP.

Table 5.1: DP applications developed in recent years.

survey question. If there are k possible answers, the bit array needs to consist of k bits. If the user gives answer i, the bit b_i must be set to 1. This approach works for a finite number k of possible answers, but already when k is large, the size of the bit array increases and the space usage gets inefficient. For infinitely many possible answers, the approach fails completely. A solution to these issues is the use of *Bloom filters*.

5.1.1.1 Bloom Filters

A Bloom filter [15] is a probabilistic data structure that can be used to efficiently determine whether a data element is member of a dataset or not. Bloom filters are implemented by a simple bit array with N individually addressable bits. This makes the data structure highly space-efficient. Initially, every bit in the Bloom filter is set to 0. To store data elements, *hash functions* are used.

In general, hash functions are functions that take data strings of arbitrary size and compress them into strings of fixed size. The classic use of hash functions is as a way to achieve expected constant lookup time for data elements. This is done by means of so-called hash tables. For a hash function H with $\mathrm{Im}(H) = \{0, 1, \ldots, N-1\}$ the hash table consists of N entries. A data element v is stored in the cell with address $H(v)$. In order to retrieve v, it suffices to compute $H(v)$ and probe that entry in the table. As N is usually smaller than the domain of the data elements, so-called *collisions* can occur. A collision is a pair of distinct data elements v and v' for which $H(v) = H(v')$ [104]. In classical hash tables, there are various ways to account for collisions, see [32].

In Bloom filters, hash functions are used to compute addresses for a given data element. But instead of storing the element in the table, only the bit in the bit array at the given address is set to 1. When several data elements are mapped to the same address, collisions occur. But as no element is stored at its address $H(v)$, when performing a lookup for v, the collisions stay undetected and an element can easily be mistaken for another. To reduce the chance of exact collisions between two elements, Bloom filters use h different hash function per element. This way, for each value v that should be stored, h different addresses $a_1, a_2, ..., a_h \in \{0, ..., N-1\}$ are calculated. Then, all h bits at the given addresses are set to 1 in the bit array representing the hash table. To check, if a certain value is present in the hash table, its addresses $a'_1, a'_2, ..., a'_h$ must be computed in the same manner as for storing a value. If all h bits in the hash table are 1, then the value is accepted.

In his paper, Bloom showed that with a larger h, the fraction of errors decreases until h gets so large that the a priori likelihood of each bit to be 1 outweighs the

effect of adding information. For Google's RAPPOR, $h = 2$ has proven to be the best choice [69].

Still, intuitively, one can see, that with this method, there is a minimum possible expected fraction of errors for any hash table size N. The errors are of the form that the setting of the hash table indicates that a certain value v is present in the table, even though it is not. This type of error is called *false positive*. The false positive rate in Bloom filters is well studied [30] and in Google's RAPPOR, it is considered to introduce additional noise into the mechanism. As the entire mechanism relies on noise introduction, this additional noise is tolerable [69].

5.1.1.2 Randomization in RAPPOR

In the RAPPOR algorithm, each user value collected for statistics is mapped into a Bloom filter directly on the user's device. The advantage that comes with the use of Bloom filters is, that in addition to numerical values, also categorical client properties, and even non-categorical values, or values from categories that cannot be enumerated ahead of time, can be collected by computing their hash values and storing them accordingly.

The RAPPOR algorithm consists of two main randomization steps, namely *permanent* and *instantaneous* randomized response, to insure the privacy of the users' Bloom filters. Figure 5.1 visualizes the data flow from the original user value v to the server.

Permanent randomized responses. For each client's value v represented in the Bloom filter B with size k and bit i, $0 \leq i \leq k$ in B, a binary reporting value B'_i is created which equals to

$$B'_i = \begin{cases} 1 & \text{with probability } \frac{1}{2}f, \\ 0 & \text{with probability } \frac{1}{2}f, \text{ and} \\ B_i & \text{with probability } 1 - f. \end{cases}$$

where f is a parameter in range $[0, 1]$ that can be tuned according to the application. It controls the longterm privacy guarantee. A high value for f causes good longterm privacy. The original Bloom filter B is discarded and never saved. Instead, the resulting B' is saved on the clients device and reused as a basis for all future reports on the user's value v. This idea is called *memorization*.

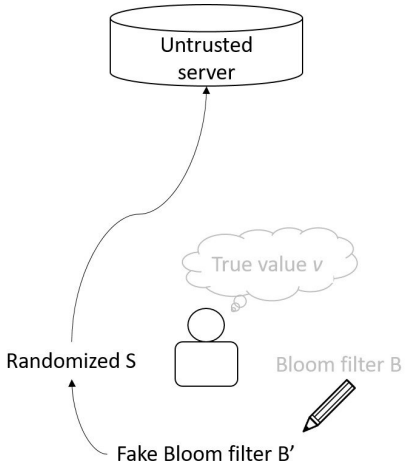


Figure 5.1: The value v represents the user's real answer. This answer is encoded in a Bloom filter B. Neither the original B nor v are ever stored on the user's device. Instead, a permanent randomized response step creates a fake Bloom filter B' from B which is saved. For every report, B' is randomized again to a bit array S, which is sent to the possibly untrusted server of the operator.

The aim of memorization is to preserve privacy against attackers who have access to multiple reports from the same user over time. Without this step, if the user had to create a new randomized Bloom filter for each report, the attacker could estimate the true responses for the bits in the noisy Bloom filter by a so-called *averaging attack*. He would just have to analyze the percentage of reports in which the bits are set to 1. With a certain amount of different B's for one user, an attacker could estimate the true value v with increasing accuracy.

In Google's RAPPOR algorithm, not every user uses the same set of hash functions to create the Bloom filters from the original data. Instead, prior to data collection, each user is randomly assigned to one of m so-called *cohorts*. Each cohort then uses a different set of hash functions. The user has to save to which cohort he belongs and send this number together with his reports, to enable analyses.

This measure has to be taken, because in Bloom filters several values map to the same bit which causes collisions. Thus, it might be, that a specific set of hash functions produces very close results for two values, making them hardly distinguishable when performing the analyses. Another set of hash functions might produce better distinguishable results for these two values, but close ones for other pairs. By using different hash functions per cohort, the chance of accidental collision for two values across all users is greatly reduced, leading to an improvement of the false positive rate in the analyses. The authors emphasize that the number m

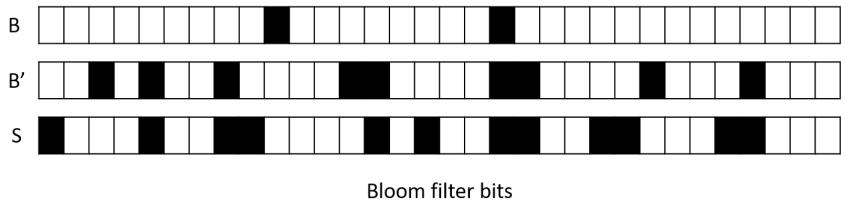


Figure 5.2: An example of the randomization process of an original Bloom filter B to the bit array S that is reported to the server with $f = 0.5$, $q = 0.75$, and $p = 0.25$. B is produced from value v by a mapping with two hash functions, resulting in 2 bits being set to 1. As a result of the randomization from permanent and instantaneous randomized response, 9 bits in B', and 12 bits in S, are set to 1.

of cohorts should be chosen carefully, because if it is too small, collisions are still quite likely, whereas, if it is too large, each individual cohort provides insufficient signal in comparison to the noise introduced, due to its small sample size [69].

Instantaneous randomized responses. Every time a report is sent to the server, a new bit array S of size k is allocated and initialized to 0. Then, each bit i in S is set with probability

$$Pr(S_i = 1) = \begin{cases} q & \text{if } B'_i = 1 \text{ and} \\ p & \text{if } B'_i = 0. \end{cases}$$

The instantaneous randomized response has several important functions. Instead of reporting B', for every request a different noisy version of it is sent to the server. This alteration makes it more difficult to track a specific client based on B', which could otherwise be used as a unique identifier over time. It also provides stronger short-term privacy guarantees, as more noise is added to the report. Figure 5.2 shows an example of how the randomization steps transform the original Bloom filter B that contains the mapping of the users value v.

5.1.2 Level of DP

To calculate the level of privacy for the RAPPOR algorithm, it is necessary to analyze both randomized response steps individually. Due to sequential composition (Theorem 3.5), the resulting values for ϵ add up to the total level of privacy.

In the RAPPOR paper [69], the authors show that the permanent randomized response satisfies ϵ_1-DP with

$$\epsilon_1 = 2h \log\left(\frac{1 - \frac{1}{2}f}{\frac{1}{2}f}\right).$$

The instantaneous randomized response satisfies ϵ_2-DP with

$$\epsilon_2 = h \log\left(\frac{q^*(1-p^*)}{p^*(1-q^*)}\right)$$

where q^* is the probability of observing 1 in S when the underlying Bloom filter bit was set, given by

$$q^* = Pr(S_i = 1|b_i = 1) = \frac{1}{2}f(p+q) + (1-f)q$$

and p^* the probability of observing 1 in S when the underlying Bloom filter bit was not set, given by

$$p^* = Pr(S_i = 1|b_i = 0) = \frac{1}{2}f(p+q) + (1-f)p.$$

The total level of privacy ϵ equals $\epsilon_1 + \epsilon_2$. With larger values for h, the level of privacy degrades whereas with f approaching one, privacy improves as this introduces more random into the result.

5.1.3 Data Analyses

To perform analyses on the user data, the noise has to be separated from the signal, the real user data. This noise extraction is performed per cohort, and after the analyses, the results from each cohort are combined. The true probability of a specific bit t_i being set in the real user data can be calculated from the observed probability c_i of this bit being set.

We know that, due to the protocol

$$c_i = \frac{f}{2}q + \frac{f}{2}p + (1-f)t_i q + (1-f)(1-t_i)p$$

	a_1	a_2	a_3
b_1	1	1	0
b_2	1	0	1
b_3	0	1	1

Figure 5.3: With three possible answers a_1, a_2, a_3, a Bloom filter size of $k = 3$, and 2 hash functions per Bloom filter, the matrix X could have the following shape. Each column has $h = 2$ ones and the ones represent which bits are set to encode the answer a_i. E.g., answer a_1 is encoded by bits b_1 and b_2 being set.

which can be solved to t_i as follows

$$
\begin{aligned}
\Leftrightarrow \quad & c_i - \frac{f}{2}q - \frac{f}{2}p = (1-f)(t_i q + (1-t_i)p) \\
\Leftrightarrow \quad & \frac{c_i - \frac{f}{2}q - \frac{f}{2}p}{(1-f)} = t_i q + p - t_i p \\
\Leftrightarrow \quad & \frac{c_i - \frac{f}{2}q - \frac{f}{2}p}{(1-f)} - \frac{p(1-f)}{(1-f)} = t_i(q-p) \\
\Leftrightarrow \quad & \frac{c_i - \frac{f}{2}q - \frac{f}{2}p - p + fp}{(1-f)(q-p)} - = t_i \\
\Leftrightarrow \quad & \frac{c_i - \frac{f}{2}q + \frac{f}{2}p - p}{(1-f)(q-p)} - = t_i.
\end{aligned}
$$

To obtain the counts for the possible user answers from the counts of the bits, [69] proposed the following procedure. We can multiply the probabilities t_i with N, the number of users in the data sample, to obtain the estimate of the true counts $\hat{t}_i$ for each bit. With these estimated counts, we can construct a vector $\vec{y}$ of $\hat{t}_i$s, with $i \in [k]$.

We can also construct a matrix X of size $k \times M$, where M is the number of candidate answers that the user could have given and k is the number of bits in the Bloom filters. This matrix encodes in every row to which Bloom filter bits the corresponding answer is mapped. These entries are set to 1 in the matrix, the other entries are set to 0, resulting in a sparse matrix. Figure 5.3 shows an example for a matrix X within one cohort. Due to the small size of the matrix in the example (3×3), it is not spare. In reality, however, it is sparse, as the number of possible answers and bits in the Bloom filters are much larger than the number of hash functions.

With vector $\vec{y}$ and matrix X, we end up with the equation system

$$\vec{y} = X\vec{\omega}.$$

The vector $\vec{\omega}$ is unknown. It represents the counts of occurrences for each possible answer. Take as an example matrix X from Figure 5.3. And let the estimates of counts be

$$\vec{y} = \begin{pmatrix} 6000 \\ 7000 \\ 5000 \end{pmatrix}.$$

We then need to calculate the vector $\vec{\omega}$ from the linear equation system

$$\begin{pmatrix} 6000 \\ 7000 \\ 5000 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}.$$

This yields $\omega_1 = 4000$, $\omega_2 = 2000$, and $\omega_3 = 3000$, and thus, the number of times that answer a_1, a_2, and a_3 were given is restored from the noisy user answers.

In reality, calculating $\vec{\omega}$ analytically might be impossible due to fact that $\vec{y}$ is only a reconstruction of the original user data from the noisy answers. Thus, it might still contain noise such that no $\vec{\omega}$ can solve the equation. Instead, we can use linear regression [119] to approximate $\vec{\omega}$. This regression should not allow negative numbers in $\vec{\omega}$, as these would represent negative counts for certain answers. Furthermore, to prevent the model from finding coefficients that are not significantly different from zero and, thus, not distinguishable from noise, some form of *regularization* should be performed. Regularization is a concept to prevent the model from overfitting to a dataset, i.e., learning the dataset by heart without generalizing enough. This is problematic because without sufficient generalization, the model also learns the noise in the dataset and not only the true data distribution. Regularization discourages learning too complex models and thereby helps to capture only the true data distribution [14].

In the Google RAPPOR paper [69], the authors propose performing two regressions on the data to approximate $\vec{\omega}$, namely

1. *lasso regression* [154] to select candidate answers with non-zero coefficients and

2. *regular least squares regression* [119] using the selected candidates.

Lasso stands for *least absolute shrinkage and selection operator* and is a regression analysis method that performs variable selection and regularization at once with the aim of enhancing the accuracy of the resulting model and making it better interpretable. Lasso forces the sum of the absolute value of the regression coefficients to be less than a fixed value. This penalizes very large coefficients to prevent overfitting and forces certain coefficients to be set to zero, effectively choosing a simpler model that does not include those coefficients. It, thus, achieves the regularization in the model.

A possible reason to use another linear regression after the lasso regression, and not to just use the estimates of the counts for possible answers produced by the lasso regression might be to separate the model selection process from the actual regression. Moreover, by performing the two regressions after each other, the estimates of the second regression are more accurate. This is, because the linear regression is performed only on the candidate answers that are kept after the regularization and not on all possible candidate answers [83]. After performing the regular least squares regression, only the counts that are statistically significant are kept.

[71] proposed a different decoding algorithm for the original RAPPOR mechanism that expands it by two additional features. The first feature consists of the possibility to study association between multiple attributes. This accounts for the fact that attributes are often more meaningful in association with other attributes. The second feature enables to estimate counts over unknown dictionaries. This is practical because in many scenarios, data elements are not known in advance to the analysis.

5.1.4 Application

The development of the RAPPOR algorithm was motivated by the need of cloud service operators to collect up-to-date statistics about the activity of their users and their client-side software [69]. Currently, the most popular application of the algorithm is its deployment in Google's Chrome web browser where it serves to collect data about Chrome users [31].

In this use case, the advantage of using Bloom filters comes into place, because the possible user answers to analysis questions consist of URLs. URLs are non-categorical data, very similar to random stings. Furthermore, considering all pages and their sub pages, there is a very large number of possible answers. For this reason and for the reason of new URLs being created at all times, the answers can not be predicted very precisely by the operator.

For the Chrome web browser 159 report metrics are specified, 47 of which are currently deprecated [31]. With the reported data, probabilistic information about Chrome users' browsing patterns, behaviors, preferred homepages, search engines, etc. can be analyzed. This offers business insights to the operator, but can also be used to prevent security risks: homepages and search engines are often targeted by malicious software and changed without the users' consent. To detect this, it is crucial to understand the distribution of these settings on a large number of Chrome installations [69].

Apart from the reporting metrics, the Chrome web browser implementation also defines two different levels of noise: In *normal noise*, $f = 0.5$, $q = 0.75$, and $p = 0.25$, and in *sparse noise*, $f = 0.25$, $q = 0.75$, and $p = 0.25$ [31]. For each reporting metric, a separate privacy budget is accounted for.

5.1.5 Critique

There are several drawbacks and limitations of the RAPPOR algorithm, that need to be analyzed.

First of all, the memorization only effectively provides longterm privacy if the true user values do not change over time, or change in an uncorrelated fashion. When the true values in the consecutive reports are temporally correlated, DP guarantees deviate from their normal levels and become weaker. To deal with this problem, the operator would have to stop collection after a certain number of reports or increase the noise level exponentially [69].

Correlation is also the largest problem in the application of the RAPPOR algorithm in the Chrome Web browser. When analyzing the metrics defined by [31], one can easily see, that many of these metrics are correlated. Take as an example the metric `AppBanner.NativeApp.Shown`, that reports the URL of a site that displayed an app banner on an android device and `AppBanner.NativeApp.Dismissed`, that reports the URL of a site on which the user explicitly dismissed the app banner. However, to dismiss a banner, the banner first has to be shown. Thus, if a user dismisses a banner, two separate metrics, having their own privacy budget, report the same information, namely which site opened a banner. Hence, the information that "User A visited site B" is more vulnerable to disclosure than it should be which shows that Google seems not to have implemented the system as differentially private as possible [37].

Another possible drawback of the original RAPPOR algorithm is that the candidate answers for analyses must be known in advance to construct the matrix X. In general, but also in the example of the Chrome web browser, this might be very

difficult. This can lead to missing some important responses. However, without loss of privacy, RAPPOR analyses can be re-run on a collection of responses to consider new strings and cases missed without the need to re-run the data collection step [69]. Thus, this drawback can be mitigated by running several analyses. Furthermore, the expansion proposed by [71] allows analyses on unknown candidate answers on the expense of increased computational complexity.

The last and possibly most difficult drawback to deal with is that operators could use the fact that users generally have multiple accounts and multiple devices. The problem with this is that the users may unknowingly participate multiple times in a report, creating multiple permanent randomized responses for the same answers. This outruns the advantages of memorization and may allow the operators to learn more information about a user than specified by ϵ. Identifying the same user over several devices is very difficult, however the operator could mitigate the problem by running collections only per account and sharing the same permanent randomized response over all devices using this account [69].

5.2 Apple

For Apple, as well as for Google, gaining insight into properties of their overall user population is crucial to improve their systems. To perform analyses to identify, for example, popular emojis or popular health data types, in a privacy preserving manner, Apple introduced three DP algorithms similar to Google's RAPPOR. The DP system was launched for the first time in macOS Sierra and iOS 10 [5]. In the following section, we will present the system architecture in which Apple deploys DP. Afterwards, we will depict the algorithms that Apple has developed to collect and analyze user data, together with their underlying data structures and their levels of privacy. At the end, we will show the use cases in which Apple applies the algorithms and discuss possible drawbacks of the system.

5.2.1 System Architecture

The system that Apple has deployed to perform privacy preserving data analyses is, as well as the Google RAPPOR algorithm, a local DP approach to avoid the need of a trusted data collector [6]. Its structure is outlined in Figure 5.4. Within this system architecture, Apple uses three algorithms that are comparable to Google's RAPPOR algorithm. Each of the three algorithms that are depicted more in detail in Section 5.2.2 consists of a client-side algorithm that coordinates the data collection

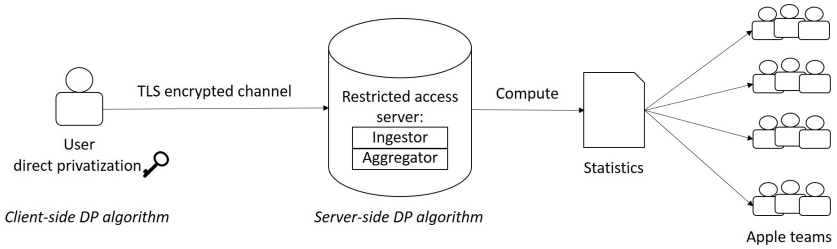


Figure 5.4: The structure of the system architecture used by Apple for their privacy preserving data analysis. The data on the user side is directly randomized and transmitted to Apple's restricted access servers once per day over a TLS encrypted channel. On the server, the *Ingestor* collects all incoming user data, separates the records per use case, removes the meta data and permutes the rows. The *Aggregator* uses the DP algorithms to generate privacy preserving histograms. These are the basis for the statistics that are distributed to the different Apple teams that are concerned by them.

on the clients' devices and a server-side algorithm that runs on Apple's restricted access servers and is responsible to perform the data analyses.

On the User's Device

In the system, a *user* has the possibility to decide whether he wants to contribute private reports about his data for analyses or not. He can change his decision at any point of time in the system settings. When a user decides to participate with his data, the collection of events on his device begins. Every event (e.g. typing an emoji on the keyboard) is directly privatized by the respective client-side algorithm and the privatized record is stored without meta data, like a device ID or a timestamp. After a certain delay, the system samples records for transmission at random from the stored ones. The transmission of privatized user records happens once per day over a TLS encrypted channel and also without any device identifiers attached to the data. The number of submissions per use case is restricted.

Apple's Restricted Access Server

After the privatized user data arrives at one of Apple's *restricted access servers*, two processing steps are performed.

1. *Ingestion.* During the ingestion step, any possible identifiers associated with the data records, like timestamp of arrival and IP identifiers, are discarded, as well as the association between multiple records. The records are then separated based on their use case and for each use case, the ordering of the privatized records is permuted randomly. The resulting data batches per use case are forwarded to the next step.

2. *Aggregation.* In the aggregation step, Apple's DP algorithms are used to generate differentially private histograms over the counts of certain events within the use cases. The data from multiple use cases is never combined. In the histograms, only events with counts over a certain threshold $\mathcal{T}$ are kept.

Based on the outcome of the aggregation step, relevant statistics can be calculated and distributed within the company.

5.2.2 Algorithms

To compute the histograms, Apple proposed three local DP algorithms for frequency estimation of a dataset, namely Count Mean Sketch *(CMS)*, Hadamard Count Mean Sketch *(HCMS)*, and Sequence Fragment Puzzle *(SFP)*. The first two operate on known dictionary settings. This means, that the elements, whose counts need to be estimated, have to be known prior to analysis. This setting is the same as within the original RAPPOR algorithm. The latter one operates on unknown dictionary settings, thus, it is not necessary to know beforehand for which elements the frequencies are to be determined [6].

All three algorithms rely heavily on a data structure called *count-min sketch* and on *count sketch* algorithms [33] performed on it.

Count-Min Sketch

The count-min sketch is a probabilistic data structure that can be used as a frequency table of events coming from a data stream. It is very similar to the Bloom filter data structure. It also uses hash functions to map from events to frequencies, and similar to Bloom filters, the hash table size can be reduced at the expense of overcounting some events due to hash collisions.

The actual data structure is a two-dimensional matrix M, called sketch, with k rows and m columns. The two parameters k and m have to be fixed before the sketch is created and determine its time and space needs, as well as the resulting probability

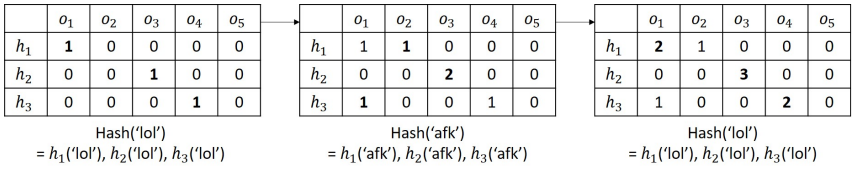


Figure 5.5: The example of adding three user-defined acronyms to a min-sketch matrix with three hash functions. First, the acronym "lol" is inserted, then "afk", which produces a collision at the second hash function, and then "lol" is added again. To estimate the count of "afk", one needs to calculate $\min\big((h_1, o_2), (h_2, o_3), (h_3, o_1)\big) = \min(1, 3, 1) = 1$.

of errors in the frequency queries [33]. Assuming that $\mathcal{I}$ is the set of all possible inputs. Associated to each of the k rows is a hash function $h_j : \mathcal{I} \to \mathcal{M}$ for all $j \in [k]$ with $1 \leq j \leq k$ and co-domain $\mathcal{M} = \{1, \ldots, m\}$, hence $|\mathcal{M}| = m$. Additionally, all k hash functions need to be pairwise independent.

Initially, all entries of sketch M must be set to zero. To add an event of type i to the data structure, for each row j, apply the corresponding hash function $h_j(i)$ to obtain a column index and increment $M_{j,h_j(i)}$ by one [33]. Figure 5.5 depicts an example of adding elements to a count-min sketch.

To query the frequency of an element with type i, the estimate count is given by the smallest value in the table for i, namely

$$\hat{a}_i = \min_j M_{j,h_j(i)}.$$

Intuitively, we can understand this in a way, that every row is a counter for the event i. Due to hash collisions, these counters are noisy. With the min-sketch approach, we select the counter with the least noise. But still, by doing so, for each i it for the true frequency that it is less or equal to the estimated frequency, i.e., $a_i \leq \hat{a}_i$. Hence, in every count, the frequency of the elements is overestimated which introduces a *bias*.

To deal with this problem, [43] proposed subtracting the expected bias from the counts leading to

$$b_j = M_{j,h_j(i)} - \frac{n - M_{j,h_j(i)}}{m - 1} \tag{5.1}$$

with n being the total number of events, and then calculating $\hat{a}_i$ as the mean over all b_j's.

$$\hat{a}_i = \frac{1}{k} \sum_{j=1}^{k} b_j. \tag{5.2}$$

This can be referred to as *mean sketch* algorithm.

To understand Equation (5.1), we can think of it in the following way: the counter $M_{j,h_j(i)}$ consists of the true counts for event i and additional noise introduced by the influence of the other events, that might, due to hash collisions be mapped to the same matrix entry. To estimate how big the influence of these other events on the counter for event i is, we can calculate their average influence to every other counter of the $m-1$ counters in that row by $(n - M_{j,h_j(i)})/(m-1)$. Assuming that they have the same influence to the counter for i, we can subtract this noise from the count to de-bias the frequency.

5.2.2.1 Count Mean Sketch

Having understood the count-min sketch data structure, we can analyze the private count mean sketch algorithm proposed by Apple. It is composed by a client-side algorithm that is responsible for the randomization of the original user data and a server-side algorithm that aggregates the privatized data and computes the estimates for the true counts from the noisy data.

Given ϵ, the number k of hash functions, the size m of their range, and a dictionary with elements from our data distribution $\mathcal{D}' \subseteq \mathcal{D}$. The algorithm performs the following steps.

1. Select a set of k three-wise independent hash functions mapping $\mathcal{D}$ to $[m]$ uniformly at random. This yields $\mathcal{H} = \{h_1, ..., h_k\}$.

2. Pass $\mathcal{H}$ and ϵ to the client-side algorithm, that is performed on a data element d, and collect the randomized user data.

3. Aggregate the randomized user data in a sketch matrix M.

4. Use the server-side algorithm to determine the estimates for the frequencies of the elements $d \in \mathcal{D}'$ from M.

5. Output the histogram of frequencies.

Client-Side Algorithm. Algorithm 5.1 depicts the procedure on the client's side. For the data element d that the client wants to report (e.g., the event that a certain emoji was used), one specific hash function is sampled at random from $\mathcal{H}$. In the concrete implementation, the data is encoded using variations of a SHA-256 hash [5]. Then, vector $\vec{v}$ is created such that it contains 1 at index $h_j(d)$ and -1 at every other index. Then, the vector is randomized by flipping each bit with probability $\frac{1}{1+e^{\epsilon/2}}$. With probability $\left(1 - \frac{1}{1+e^{\epsilon/2}}\right) = \frac{e^{\epsilon/2}}{1+e^{\epsilon/2}}$, the bit stays the same. The randomized vector together with the index j is then transmitted to the server. There, it is added to a row in the sketch matrix.

Algorithm 5.1 Client-Side CMS algorithm, adapted from [6, Algorithm 2]

function CLIENT-SIDE CMS($d \in \mathcal{D}$, ϵ, $\mathcal{H}$)
  $j \leftarrow$ sampled uniformly at random from $[k]$, to select $h_j \in \mathcal{H}$.
  $\vec{v} \leftarrow (-1, -1, \ldots, -1) \in \mathbb{R}^m$ with -1 at every component.
  Set $\vec{v}_{h_j(d)} \leftarrow 1$.
  $\tilde{\vec{v}} \leftarrow \vec{v}$, where sign of every element of $\vec{v}$ is flipped with probability $\frac{1}{1+e^{\epsilon/2}}$.
  return $\left(\tilde{\vec{v}},\ \text{index } j\right)$
end function

Compute Sketch Matrix On server side, the incoming privatized data records are combined into a sketch matrix M by Algorithm 5.2.

Algorithm 5.2 Compute Sketch Matrix, adapted from [6, Algorithm 3]

1: **function** COMPUTE SKETCH MATRIX($D = \{(\tilde{\vec{v}}^{(1)}, j^{(1)}), ..., (\tilde{\vec{v}}^{(n)}, j^{(n)})\}, \epsilon, k, m$)
2: Set $c_\epsilon \leftarrow \frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}$.
3: **for** $i \in \{1, ..., n\}$ **do**
4: Set $\vec{\tilde{x}}^{(i)} \leftarrow k \cdot \left(\frac{c_\epsilon}{2}\vec{\tilde{v}}^{(i)} + \frac{1}{2}\vec{1}\right)$.
5: **end for**
6: Initialize sketch matrix $M = \{0\}^{k \times m}$.
7: **for** $i \in \{1, ..., n\}$ **do**
8: **for** $l \in \{1, ..., m\}$ **do**
9: $M_{j^{(i)},l} \leftarrow M_{j^{(i)},l} + \vec{\tilde{x}}_l^{(i)}$.
10: **end for**
11: **end for**
12: **return** M.
13: **end function**

Understanding what happens in line 4 of Algorithm 5.2 as described in [6] is not very intuitive. To understand better, why the algorithm really outputs a sketch matrix with the estimated counts, it helps to consider several points individually.

1. *The multiplication by k.* As we have seen in the centralized min-sketch data structure, adding an event causes k counters, corresponding each to a row in the sketch matrix, to be incremented by 1. In the distributed setting of the mean sketch algorithm, each event only causes one counter to be incremented. To maintain the properties of the data structure, the vector which is added is multiplied by k. Thus, instead of incrementing k counters by a value, one counter is incremented by k times this value. When adding a large number of events n, this corresponds to the outcome of the centralized version, as the rows j are sampled uniformly at random.

2. *The assignment of* $(\frac{c_\epsilon}{2}\vec{v}^{(i)} + \frac{1}{2}\vec{1})$ *to* $\vec{x}^{(i)}$. To understand the implications of this term, it helps to consider the values of c_ϵ. We can see that $\lim_{\epsilon\to\infty} c_\epsilon = 1$ and $\lim_{\epsilon\to 0} c_\epsilon = \infty$. Thus, with large ϵ (weak privacy), this term corresponds to $\approx (\frac{1}{2}\vec{v}^{(i)} + \frac{1}{2}\vec{1})$, mapping the privatized user vector $\vec{v}^{(i)} \in \{-1, 1\}^m$ to a vector $\in \{0, 1\}^m$, in which the -1 is replaced by 0 and 1 stays 1. With small ϵ (stronger privacy), the -1 and 1 are replaced by large negative or positive numbers respectively.

3. *The choice of* $c_\epsilon = \frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}$. This choice of c_ϵ is crucial to obtain a sketch matrix with the correct approximated counts. To prove this, we analyze the expected value for each matrix entry.

To argue about the choice for c_ϵ, we need to introduce some notation. Let $B \in \{-1, 1\}$ denote a random variable that is 1 with probability $\frac{e^{\epsilon/2}}{1+e^{\epsilon/2}}$. This B is used for the randomization process of the original vector $\vec{v}$ of user data, in which every bit stays the same with probability $\frac{e^{\epsilon/2}}{1+e^{\epsilon/2}}$. Furthermore, let J denote a random variable drawn uniformly at random from $[k]$. We set the encoding vector $\vec{v}^{(i)} \in \{-1, 1\}^m$ to be -1 everywhere, except at position $h_J(d^{(i)})$, where $d^{(i)}$ is the data entry for i. To prevent the need for too many superscripts, we denote the $h_j(d)^{th}$ index of $\vec{v}^{(i)}$ as $v^{(i)}[h_j(d)]$. The entry of every cell $(j, h_j(d))$ in the sketch matrix for data record i can be written as

$$Y_j^{(i)}(d) = k\left(\frac{c_\epsilon B v^{(i)}[h_j(d)] + 1]}{2}\right)\mathbb{1}_{\{J=j\}}, \tag{5.3}$$

where $Y_j^{(i)}(d)$ is non-zero only if and only if $J = j$ [6]. This is expressed by $\mathbb{1}_{\{J=j\}}$.

Note. If i^{th} data entry $d^{(i)} = d$, then the cell content $v^{(i)}[h_j(d)] = 1$. This is because in this cell, the counter was incremented by one with probability 1 according to the protocol for the construction of the sketch matrix. If $d^{(i)} \neq d$, then $v^{(i)}[h_j(d)] = 1$ with probability $1/m$. This is because the hash function could wrongly map another data element to the same value due to hash collisions. The probability $1/m$

results from the fact that $\mathcal{H}$ is chosen uniformly at random from the three-wise independent hash functions with ranges of size m.

We can now consider the expected value for the cells in the sketch matrix for both cases, $d^{(i)} = d$ and $d^{(i)} \neq d$, and use the fact that $\mathbb{E}[B] = \frac{1}{c_\epsilon}$.

Case 1: $d^{(i)} = d$

$$\begin{aligned}\mathbb{E}[Y_i^{(i)}(d)] &= k\mathbb{E}\left[\frac{c_\epsilon B v^{(i)}[h_j(d)] + 1]}{2} | J = j\right] Pr[J = j] \\ &= \mathbb{E}\left[\frac{c_\epsilon B v^{(i)}[h_j(d)] + 1]}{2}\right] \\ &= \mathbb{E}\left[\frac{c_\epsilon B \cdot 1 + 1}{2}\right] = 1\end{aligned}$$

Case 2: $d^{(i)} \neq d$

$$\begin{aligned}\mathbb{E}[Y_i^{(i)}(d)] &= k\mathbb{E}\left[\frac{c_\epsilon B v^{(i)}[h_j(d)] + 1]}{2} | J = j\right] Pr[J = j] \\ &= \left(1 - \frac{1}{m}\right)\mathbb{E}\left[\frac{c_\epsilon B \cdot (-1) + 1}{2}\right] + \frac{1}{m}\mathbb{E}\left[\frac{c_\epsilon B \cdot 1 + 1}{2}\right] \\ &= \left(1 - \frac{1}{m}\right) \cdot 0 + \frac{1}{m} \cdot 1 = \frac{1}{m}\end{aligned}$$

This shows us that, with the given choice of c_ϵ, the expected values in the sketch matrix correspond to correct estimates for the true counts.

Server-Side Algorithm. The server-side algorithm uses the sketch matrix to estimate the count for a data element $d \in \mathcal{D}$. It does so by de-biasing the counts and averaging over the corresponding hash entries in M. The procedure is presented in Algorithm 5.3.

To understand that the estimation of the counts really produces a de-biased mean sketch as defined by Equation (5.2) on page 68, we can rearrange the equation for $\tilde{f}(d)$.

Algorithm 5.3 Server-Side CMS algorithm, adapted from [6, Algorithm 4]

1: **function** SERVER-SIDE CMS($d \in \mathcal{D}$, ϵ, $M \in \mathbb{R}^{k \times m}$, $\mathcal{H}$, number of data elements n)
2: Construct $\tilde{f} : \mathcal{D} \to \mathbb{R}$ where

$$\tilde{f}(d) = \left(\frac{m}{m-1}\right)\left(\frac{1}{k}\sum_{l=1}^{k} M_{l,h_l(d)} - \frac{n}{m}\right)$$

3: **return** $\tilde{f}(d)$
4: **end function**

$$\begin{aligned}
\tilde{f}(d) &= \left(\frac{m}{m-1}\right)\left(\frac{1}{k}\sum_{l=1}^{k} M_{l,h_l(d)} - \frac{n}{m}\right) \\
&= \frac{1}{k}\left(\sum_{l=1}^{k} \frac{M_{l,h_l(d)} \cdot m}{m-1} - \frac{nm}{m(m-1)}\right) \\
&= \frac{1}{k}\left(\sum_{l=1}^{k} \frac{M_{l,h_l(d)} \cdot m - 1}{m-1} + \frac{M_{l,h_l(d)} \cdot 1}{m-1} - \frac{n}{m-1}\right) \\
&= \frac{1}{k}\left(\sum_{l=1}^{k} M_{l,h_l(d)} + \frac{M_{l,h_l(d)}}{m-1} - \frac{n}{m-1}\right) \\
&= \frac{1}{k}\left(\sum_{l=1}^{k} M_{l,h_l(d)} - \frac{n - M_{l,h_l(d)}}{m-1}\right),
\end{aligned}$$

which is exactly the formulation of the de-biased mean sketch as given by Equations (5.1) and (5.2) on page 67.

5.2.2.2 Hadamard Count Mean Sketch

As stated above, the number of errors for the estimated counts of events depends on the size of the sketch matrix M. The larger M is, the smaller is the error rate. To improve the accuracy of their estimates even in large datasets or datasets from a wide-spread data distribution, Apple decided to increase the size of M by increasing k and m. But this would lead to the users having to report larger vectors to the server, resulting in a consumption of their bandwidth. To avoid this situation, Apple developed the HCMS algorithms which still forms a $k \times m$ sketch matrix on the server, but only needs one-bit reports from the user. This bit is created from the

vector $\vec{v}$ on the client side by performing the Hadamard transform [90]. For a more detailed description of the HCMS, see [6].

5.2.2.3 Sequence Fragment Puzzle

The previous two algorithms assume that there is a known dictionary $\mathcal{D}' \subseteq \mathcal{D}$ over which the server can enumerate to estimate the counts for all data entries $d \in D'$. However, this assumption does not always hold. An example for a scenario with an unknown dictionary is the addition of new words learned from the user input to the pre-defined dictionary. Brute-forcing the process of finding popular new words by assuming $\mathcal{D}'$ to contain all possible words of all possible lengths and enumerating over them is computationally too complex. It furthermore yields too many false positives due to the addition of noise.

As a solution, Apple introduced the SFP algorithm [6]. It is presented in the following for strings of length 10 (longer strings are truncated and shorter ones are padded with spaces), but the approach can easily be generalized to arbitrary-length strings. The algorithm is composed, as well as the previous algorithms, by a client-side algorithm that is responsible for the randomization of the original user data and a server-side algorithm that aggregates the privatized data and computes the estimates for the true counts from the noisy data.

Given a dataset $d^{(0)}, ..., d^{(n)} \in \mathcal{D}^{(n)}$, two privacy budgets ϵ, ϵ', the numbers k, k' of hash functions, the sizes m, m' of their ranges, a hash function $h : \mathcal{D} \to [256]$, and a threshold $\mathcal{T}$. The algorithm performs the following steps.

1. Select a set of k three-wise independent hash functions mapping $\mathcal{D}$ to $[m]$ uniformly at random. This yields $\mathcal{H} = \{h_1, ..., h_k\}$.

2. Select a set of k' three-wise independent hashes mapping $\mathcal{D}$ to $[m']$ uniformly at random. This yields $\mathcal{H}' = \{h'_1, ..., h'_k\}$.

3. Pass $\mathcal{H}, \mathcal{H}'$, hash function h, and ϵ, ϵ' to the client-side algorithm that is performed on a user string d – which we are referring to as $\vec{s}$ to indicate that it is a vector of characters – and collect the randomized user data.

4. Aggregate the randomized user data.

5. Use the server-side algorithm to determine a dictionary $\mathcal{D}$ and frequencies for all data elements $d \in \mathcal{D}$.

Client-Side Algorithm. Algorithm 5.4 depicts the procedure on the client side. $\vec{s}[i : j]$ denotes the characters in $\vec{s}$ starting at index i and ending at index j. The algorithm first calculates a small range hash function h on the string $\vec{s}$. Then, a two-character sequence from the stings $\vec{s}$ is sampled uniformly at random from the odd indices $l \in \{1, 3, 5, 7, 9\}$ to construct a substring $\vec{s}[l : l + 1]$[1]. Vector $\vec{r}$ is calculated as a concatenation of the output of the small range hash function h on $\vec{s}$ and the two-character substring. The CMS algorithm is then used to randomize $\vec{r}$ and $\vec{s}$.

Algorithm 5.4 Client-Side SFP algorithm, adapted from [6, Algorithm 9]

1: **function** CLIENT-SIDE SFP($\vec{s} \in \mathcal{D}$, (ϵ, ϵ'), $(\mathcal{H}, \mathcal{H})'$, h)
2: $l \leftarrow$ sampled uniformly at random from $\{1, 3, 5, 7, 9\}$.
3: Set $\vec{r} \leftarrow h(\vec{s}) \| \vec{s}[l : l + 1]$.
4: $\alpha \leftarrow$ Client-Side CMS($\vec{r}, \epsilon', \mathcal{H}'$)
5: $\beta \leftarrow$ Client-Side CMS($\vec{s}, \epsilon, \mathcal{H}$)
6: **return** (α, β, l).
7: **end function**

The aim of concatenating the small hash value with the two-character strings in $\vec{r}$ is to correlate the two-character fragment with the full word in which it appears.

Server-Side Algorithm. Algorithm 5.5 shows the process on server side. First, a frequency oracle is determined over all new words that the users have used (line 3). Then, for every index l, another frequency oracle is calculated over the randomized vectors $\vec{r}_i$ being sampled at index l (line 7). Afterwards, every combination of $\omega \in [256]$ (possible outputs of hash function h) and all two-character strings from the given alphabet, are presented to the frequency oracles for each index l to determine which substrings appear with highest frequency at this index (line 10-13). To build the possible words, the Cartesian product of all the most frequent two-character substrings, is calculated (line 16-17). Finally, the first frequency oracle determines the frequencies for the words resulting from this process.

Line 12-13 show that the algorithm iterates over all possible two-character fragments. If these fragments were not correlated to the word they were taken out from by $h(\vec{s})$, at this point, there might be fragments that have very large counts in each position but are not part of any word, thus leading to false positives [6].

The paper [6] does not give any indication on how to choose the threshold $\mathcal{T}$.

[1] Note that it is possible to use substrings of arbitrary length, but that substrings of length 2 are used for easier exposition.

Algorithm 5.5 Server-Side SFP algorithm, adapted from [6, Algorithm 10]

function SERVER-SIDE SFP(Client submission $(a^{(i)}, \beta^{(i)}, l^{(i)})$, (ϵ, ϵ'), $(\mathcal{H}, \mathcal{H})'$, $\mathcal{T}$)
  Create sketch $M \leftarrow$ Compute Sketch Matrix$(\beta^{(1)}, ..., \beta^{(n)}; \epsilon, k, m)$.
  Create frequency oracle $\tilde{f}(\cdot.)$ with Server-Side CMS$(\cdot; M, \epsilon, \mathcal{H})$.
  for $l \in \{1, 3, 5, 7, 9\}$ **do**
    Denote $S_l \subseteq [n]$ as the set of client records that have selected index l.
    Create sketch $M_l \leftarrow$ Compute Sketch Matrix$(a^{(i)} : i \in S_l; \epsilon', k', m')$.
    Use Server-Side CMS$(\cdot; M_l, \epsilon', \mathcal{H}')$ to create a frequency oracle $\tilde{f}_l(\cdot)$
  end for
  Initialize a dictionary $\mathcal{D} \leftarrow \emptyset$.
  for $l \in \{1, 3, 5, 7, 9\}$ **do**
    Initialize $\mathcal{Q}_l \leftarrow \emptyset$.
    Fill $\mathcal{Q}_l$ with $\mathcal{T}$ tuples $(\omega \| \vec{s})$ with the largest counts $\tilde{f}_l(\omega \| \vec{s})$
      for $\vec{s} \in \Omega^2, \omega \in [256]$ for alphabet Ω.
  end for
  for $\omega \in [256]$ **do**
    Form the Cartesian product of two-character strings in
      $\mathcal{Q}(\omega) = \{q_1 \| ... \| q_9 : \omega \| q_l \in \mathcal{Q}_l \text{ for } l \in \{1, 3, 5, 7, 9\}\}$.
  end for
  for $\vec{s} \in \mathcal{Q}(\omega)$ **do**
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\vec{s}\}$.
  end for
  return Dictionary $\mathcal{D}$ and frequencies $\tilde{f}(d)$ for $d \in \mathcal{D}$.
end function

5.2.3 Level of DP

In their paper, Apple shows that CMS and HCSM are ϵ-differentially private algorithms. Due to the composition, SFP is an $(\epsilon + \epsilon')$-differentially private algorithm. For a proof see [6].

5.2.4 Applications

Currently, Apple uses their DP algorithms to perform analyses of the following features [5, 6]

1. *QuickType Suggestions* to identify most popular words to suggest to the user while he is typing.

2. *Discovering New Words* to learn new words from user inputs that are not present in the lexicons.

3. *Emoji Suggestions* to suggest emojis to the user based on what he is typing.

4. *Discovering Popular Emojis* to identify the most popular emojis across the user base.

5. *Lookup Hints* to provide lookup suggestions to a user.

6. *Safari Energy Draining Domains* to detect websites that cause high energy drain due to CPU usage.

7. *Safari Autoplay Intent Detection (macOS High Sierra)* to infer default auto-play policies for websites based on user behavior.

8. *Safari Crashing Domains (iOS 11)* to detect crashing websites.

9. *Health Type Usage (iOS 10.2)* to learn popular health types for future improvement of the Health app. No user data is included, only which health data types are edited by the user.

For each of these features, Apple grants a certain privacy budget ϵ per day and and limits the number of submissions. Table 5.2 provides an overview over the properties for each feature as taken from [5, 6]. If no information is published about a certain value, the cell is left empty.

Some choices of feature properties are worth to be analyzed more in detail. The first one is the choice of ϵ, together with the number of data donations per user per day. It seems reasonable to treat sensitive user health data more carefully (by assigning a smaller ϵ to them) than websites visited with Safari. The second one is that, to report Safari domains causing high energy consumption or providing Auto-play options, Apple does not use the SFP algorithm that would make it possible to discover properties of domains not listed in a pre-defined dictionary. Instead, they rely on a dictionary containing 250000 domains. This is a similar approach to the RAPPOR algorithm, in which Google also performs their analyses only on a set of pre-defined domains. Possible reasons for Apple to choose CMS over the more flexible SFP could be that the information about the 250000 most popular websites is sufficient for these use cases, or that the SFP algorithm has a higher computational complexity than the CMS on strings with arbitrary lengths. Another interesting choice when it comes to reporting about Safari domains is that m for the high energy consuming domains is much larger than for Auto-play intent domains, whereas its k is much smaller. This justifies why the first one uses HCMS

Feature	Algorithm	ϵ	m	k	p	d
QuickType Suggestions		8				2
Discovering New Words	SFP	8	1024	2048	/	
Emoji Suggestions		4				1
Discovering Popular Emojis	CMS	4	1024	65536	2600	
Lookup Hints		4				2
Safari Energy Draining Domains	HCMS	4	32768	1024	250000	2
Safari Auto-play Intent	CMS	8	1024	65536	250000	2
Safari Crashing Domains						2
Health Type Usage	CMS	2	256	65536		1

Table 5.2: Overview of feature properties, where p denotes the size of the underlying dictionary, if counts for a known dictionary are to be estimated, and d denotes the number of donations per day. Also the privacy budget ϵ is granted on daily basis. Empty cells suggest that Apple has not published any data about that feature property.

and the latter CMS. Even though the resulting sketch matrix about the Auto-play feature is larger than the one for the energy consumption, the rows (which the users report) are much smaller (1024×1) vs. (32768×1).

5.2.5 Critique

There has been some severe critique on Apple's implementation and deployment of DP.

The first critique was that, when freshly deployed, Apple did not provide any information about the implementation or the choice of the privacy parameters of their DP mechanisms. The only hints about the algorithms used were three patents taken out by Apple employees at that time [151–153]. It took researchers several month to explore the system and estimate the privacy budgets [149]. Only when these results were presented did Apple publish about their DP system [6].

The researchers did also find that Apple awards a fresh privacy budget to each user and metric every day. Therefore, the privacy budget is increased by the daily ϵ every 24 hours. This yields two severe problems. First, due to composition, the

daily privacy budgets add up, resulting in an unbounded privacy loss over the user's lifetime. Second, if the privacy budget is not consumed daily, e.g., if a user does not create any data records, it accumulates over several days. The next time the user creates data for submission, the whole accumulated budget can be spent in one session, thereby, causing large privacy loss.

Another point of criticism was that Apple performs the analyses on data batches containing three month of user data. As every user is allowed to donate new data every day, there are weeks worth of user data in a single analysis, leading to a decrease in privacy [37].

Furthermore, as we have seen, the choice of ϵ is important to truly protect the users' privacy. According to experts, $0 \leq \epsilon \leq 1$ is a good choice for ϵ. When looking at Table 5.2, we see that Apple uses an ϵ of up to 16 ($2 \cdot 8$) for their metrics per day, thus, only yielding very weak privacy protection [78].

At last, even though Apple praises its system for implementing local DP algorithms, it seems that they undermine the advantages of a local approach: Apple collects data for the described metrics and claims that data from different metrics is never merged together. The user has no choice but trusting Apple with this claim. However, in its pure form, a local approach should supersede the need for trust in any party. It might seem that the metrics, such as health data types and the use of emojis, are uncorrelated anyways, resulting in a lower need for trust. But, just because no one can find a relation between such metrics now, does not mean that it does not exist and that someone who discovers it could use it in a malicious way.

6 Discussion

The previous chapters gave a broad overview of many aspects of DP. As a result, we understand that DP is better suited for some types of data than for others. Especially in data with strong correlations, the privacy provided by DP is worse than initially suggested. Furthermore, the practicability of DP depends heavily on the application. Whereas the application of the Laplace mechanism on data mining tasks that depend on counting queries is straightforward and can achieve reasonable results, it has proven non-trivial even for the most basic machine learning algorithms like linear regression.

The experimental part of this thesis in Chapter 4 implies that specific DP mechanisms do not achieve high or low quality results per se. The same mechanism can yield results of different quality dependent on its application. Method 1, as well as method 2, use the Laplace mechanism. In the first one, it is applied to perturb the prediction output. This leads to the addition of so large amounts of noise that the results are rendered useless for privacy preserving data analyses. In the second one, it is used to perturbed the cost function instead of the prediction output. This produces more accurate results even for smaller values of ϵ, thus, favoring privacy-preserving analyses.

The implementation also suggests that it is not trivial to transform a machine learning algorithm in a way that it preserves DP. Even method 1, the naive approach, is based on many assumptions and requires a deep understanding of the mathematical foundations of DP. Method 2, the functional mechanism, moreover, depends on a deep mathematical knowledge to understand the procedure, e.g., transforming a function into its polynomial representation. To our knowledge, so far, no libraries exist in any programming language offering DP implementations for standard machine learning tasks, thereby, overcoming these problems. This renders the application of DP for the broad mass of machine learning users impossible as they are lacking the required level of knowledge about DP.

Chapter 5 shows that the importance of privacy-preserving data analysis in the real world is so high that influential companies put a lot of effort into designing their own DP applications. But contrary to their claims about providing absolute privacy preservation for their users, their implementations have some major drawbacks. In our opinion, the largest danger that comes with this is that the users gain a

wrong feeling of security and are, therefore, more willing to share their private data without understanding the potential risks. Apple furthermore promotes that its users have the free choice to participate in the analyses with their data or not. In general, this is a good idea, however, we consider this choice problematic. This is because the mechanisms are so complex and difficult to understand that the vast majority of users can not make an *informed* choice and might therefore tend to trust in the promises of perfect privacy preservation more than they should.

Even for individuals who do understand the mechanisms, there is still the largest drawback of DP, namely the choice of ϵ. In the literature, we saw that there is no consensus about it. The implementations in Chapter 4 imply that with increasing ϵ, the accuracy of the data analyses improves. Chapter 5 shows that in real-world applications, the companies tend to prefer larger values for ϵ, probably for that exact reason. With the choice of ϵ seeming arbitrary and being left to the operator, real privacy preservation is doubtful.

In Chapter 1, we depicted the promised qualities of DP. Now, we can evaluate how well these are implemented in practice. Indeed, the value ϵ gives a concrete value to quantify privacy loss. It furthermore enables to compare different mechanisms and to define a maximum bound for tolerated privacy loss over multiple computations (composition theorem) and in groups (group privacy). In principle, this works well, however in real-world implementations, apart from the seemingly arbitrary choice of this value ϵ, there is another problem. To really ensure that the privacy loss is bounded, it would be necessary to prevent any user who has consumed his privacy budget from ever accessing the data again. It is highly questionable that such a restriction can be realized because of the difficulty of re-identifying a single user. Even in the simplest case, a database with restricted user access, after a user has consumed his privacy budget, it is very difficult to prevent him from actions such as simply creating a new user account and having a fresh privacy budget. In potentially uncontrolled data publication, this problem is even more difficult to overcome. This also challenges the last advertised property of DP, its immunity to post processing. The problem with post processing is that even if it is possible to ensure that a single user only has his limited and well-defined privacy budget, there is still no possible way to prevent that several parties collude and combine their privacy budgets or analysis results to achieve a privacy breach in a DP setting.

Lastly, it is important to point out that DP does not prevent any individual from being harmed by exhibiting some properties. Take as an example the property if someone is a smoker. If an analysis is conducted to find out about the influence of smoking on lung cancer, DP ensures to the individual that his participation in the dataset stays undetected. This is achieved by ensuring that the results of the analysis on the dataset are approximately the same whether or not the individual is in the data. This, however, also implies that the findings from the analysis are

more or less the same independent of the participation of the individual. If it is found that smoking indeed causes lung cancer and health insurances, therefore, decide to raise the insurance prices for smokers, the individual is still harmed by these findings, even if his participation in the data is dissimulated.

All in all, DP seems to be a promising concept. Its implementation is a step into the right direction on the way of performing privacy preserving data analyses. However, to really become applicable, the choice of a value for ϵ would have to become more tangible. Further research should, therefore, focus on translating ϵ into something human-understandable in the form of "If I donate my data to this dataset, with the given value for ϵ, this and that would have to happen to reveal this and that property about me." Moreover, DP versions of standard machine learning algorithms should be implemented as libraries for the common programming languages to make the use of privacy preserving analysis methods accessible for every user even if he does not possess the background knowledge about DP.

Bibliography

[1] Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep Learning with Differential Privacy". In: (2016), pp. 308–318. DOI: 10.1145/2976749.2978318. arXiv: 1607.00133 [cs, stat].

[2] Gergely Ács and Claude Castelluccia. "I Have a DREAM! (DiffeRentially privatE smArt Metering)". In: *Information Hiding*. Ed. by Tomáš Filler, Tomáš Pevný, Scott Craver, and Andrew Ker. Vol. 6958. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 118–132. ISBN: 978-3-642-24177-2 978-3-642-24178-9. DOI: 10.1007/978-3-642-24178-9_9. URL: http://link.springer.com/10.1007/978-3-642-24178-9_9 (visited on 11/09/2018).

[3] Nabil R. Adam and John C. Worthmann. "Security-Control Methods for Statistical Databases: A Comparative Study". In: *ACM Computing Surveys* 21.4 (Dec. 1, 1989), pp. 515–556. ISSN: 03600300. DOI: 10.1145/76894.76895.

[4] Charu C. Aggarwal and Philip S. Yu. "A Condensation Approach to Privacy Preserving Data Mining". In: *Advances in Database Technology - EDBT 2004*. Ed. by Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Böhm, and Elena Ferrari. Red. by Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen. Vol. 2992. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 183–199. ISBN: 978-3-540-21200-3 978-3-540-24741-8. DOI: 10.1007/978-3-540-24741-8_12. URL: http://link.springer.com/10.1007/978-3-540-24741-8_12 (visited on 11/07/2018).

[5] Differential Privacy Team Apple. *Differential Privacy*. 2016. URL: https://images.apple.com/privacy/docs/Differential_Privacy_Overview.pdf (visited on 09/07/2018).

[6] Differential Privacy Team Apple. "Learning with Privacy at Scale". In: (2017). URL: https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf (visited on 07/18/2018).

[7] Jane Bambauer, Krishnamurty Muralidhar, and Rathindra Sarathy. "Fool's Gold: An Illustrated Critique of Differential Privacy". In: 16 (2013), p. 55.

[8] R.J. Bayardo and R. Agrawal. "Data Privacy through Optimal K-Anonymization". In: *21st International Conference on Data Engineering (ICDE'05)*. 21st International Conference on Data Engineering (ICDE'05). Tokyo, Japan: IEEE, 2005, pp. 217–228. ISBN: 978-0-7695-2285-2. DOI: 10.1109/ICDE.2005.42.

[9] Donald Beaver. "Commodity-Based Cryptography". In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (1997), pp. 446–455. URL: http://delivery.acm.org/10.1145/260000/258637/p446-beaver.pdf?ip=87.77.41.246&id=258637&acc=ACTIVE%20SERVICE&key=2BA2C432AB83DA15%2EB8306ECA494DD095%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1541702515_ef0bfd9f073b5c027ee523a93c829a8e (visited on 11/08/2018).

[10] Donald Beaver. "Server-Assisted Cryptography". In: *Proceedings of the 1998 Workshop on New Security Paradigms - NSPW '98*. The 1998 Workshop. Charlottesville, Virginia, United States: ACM Press, 1998, pp. 92–106. ISBN: 978-1-58113-168-0. DOI: 10.1145/310889.310923.

[11] Leland L Beck. "A Security Mechanism for Statistical Databases". In: *ACM Transactions on Database Systems* 5.3 (Sept. 1980), p. 23.

[12] Amos Beimel, Kobbi Nissim, and Uri Stemmer. "Private Learning and Sanitization: Pure vs. Approximate Differential Privacy". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Ed. by Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim. Red. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, and Gerhard Weikum. Vol. 8096. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 363–378. ISBN: 978-3-642-40327-9 978-3-642-40328-6. DOI: 10.1007/978-3-642-40328-6_26. URL: http://link.springer.com/10.1007/978-3-642-40328-6_26 (visited on 09/07/2018).

[13] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation". In: *Proceedings of the twentieth annual ACM symposium on Theory of computing* (1988), pp. 1–10. URL: http://www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/GBW88/GBW88.pdf (visited on 11/08/2018).

[14] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information science and statistics. New York: Springer, 2006. 738 pp. ISBN: 978-0-387-31073-2.

[15] Burton H. Bloom. "Space/Time Trade-Offs in Hash Coding with Allowable Errors". In: *Communications of the ACM* 13.7 (July 1, 1970), pp. 422–426. ISSN: 00010782. DOI: 10.1145/362686.362692.

[16] Dan Bogdanov, Sven Laur, and Jan Willemson. "Sharemind: A Framework for Fast Privacy-Preserving Computations". In: *Computer Security - ESORICS 2008*. Ed. by Sushil Jajodia and Javier Lopez. Vol. 5283. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 192–206. ISBN: 978-3-540-88312-8 978-3-540-88313-5. DOI: 10.1007/978-3-540-88313-5_13. URL: http://link.springer.com/10.1007/978-3-540-88313-5_13 (visited on 11/09/2018).

[17] *Boston Housing*. URL: https://www.kaggle.com/c/boston-housing (visited on 12/24/2018).

[18] Mark Bun, Jonathan Ullman, and Salil Vadhan. "Fingerprinting Codes and the Price of Approximate Differential Privacy". In: (Nov. 13, 2013). arXiv: 1311.3158 [cs]. URL: http://arxiv.org/abs/1311.3158 (visited on 11/16/2018).

[19] Ji-Won Byun, Elisa Bertino, and Ninghui Li. "Purpose Based Access Control of Complex Data for Privacy Protection". In: *Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies - SACMAT '05*. The Tenth ACM Symposium. Stockholm, Sweden: ACM Press, 2005, p. 102. ISBN: 978-1-59593-045-3. DOI: 10.1145/1063979.1063998.

[20] Ji-Won Byun and Ninghui Li. "Purpose Based Access Control for Privacy Protection in Relational Database Systems". In: *The VLDB Journal* 17.4 (July 2008), pp. 603–619. ISSN: 1066-8888, 0949-877X. DOI: 10.1007/s00778-006-0023-0.

[21] Jianneng Cao, Panagiotis Karras, Panos Kalnis, and Kian-Lee Tan. "SABRE: A Sensitive Attribute Bucketization and REdistribution Framework for t-Closeness". In: *The VLDB Journal* 20.1 (Feb. 2011), pp. 59–81. ISSN: 1066-8888, 0949-877X. DOI: 10.1007/s00778-010-0191-9.

[22] Konstantinos Chatzikokolakis, Miguel E. Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. "Broadening the Scope of Differential Privacy Using Metrics". In: *Privacy Enhancing Technologies*. Ed. by Emiliano De Cristofaro and Matthew Wright. Red. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, and Gerhard Weikum. Vol. 7981. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 82–102. ISBN: 978-3-642-39076-0 978-3-642-39077-7. DOI: 10.1007/978-3-642-39077-7_5. URL: http://link.springer.com/10.1007/978-3-642-39077-7_5 (visited on 07/18/2018).

[23] Kamalika Chaudhuri and Claire Monteleoni. "Privacy-Preserving Logistic Regression". In: *Advances in Neural Information Processing Systems*. 2009, pp. 289–296.

[24] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. "Differentially Private Empirical Risk Minimization". In: *Journal of Machine Learning Research* 12 (2011), pp. 1069–1109.

[25] David Chaum, Claude Crépeau, and Ivan Damgard. "Multiparty Unconditionally Secure Protocols". In: *Proceedings of the twentieth annual ACM symposium on Theory of computing* (1988), pp. 11–19. URL: http://crypto.cs.mcgill.ca/~crepeau/PDF/ASPUBLISHED/CCD88A.pdf (visited on 11/08/2018).

[26] Rui Chen, Bipin C Desai, Benjamin C M Fung, and Nériah M Sossou. "Differentially Private Transit Data Publication: A Case Study on the Montreal Transportation System". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, p. 9.

[27] F.Y. Chin and G. Ozsoyoglu. "Auditing and Inference Control in Statistical Databases". In: *IEEE Transactions on Software Engineering* SE-8.6 (Nov. 1982), pp. 574–582. ISSN: 0098-5589. DOI: 10.1109/TSE.1982.236161.

[28] Francis Chin. "Security Problems on Inference Control for SUM, MAX, and MIN Queries". In: *Journal of the ACM* 33.3 (May 1, 1986), pp. 451–464. ISSN: 00045411. DOI: 10.1145/5925.5928.

[29] Francis Chin and Gultekin Ozsoyoglu. "Auditing for Secure Statistical Databases". In: *Proceedings of the ACM '81 Conference on - ACM 81*. The ACM '81 Conference. Not Known: ACM Press, 1981, pp. 53–59. ISBN: 978-0-89791-049-1. DOI: 10.1145/800175.809832.

[30] Ken Christensen, Allen Roginsky, and Miguel Jimeno. "A New Analysis of the False Positive Rate of a Bloom Filter". In: *Information Processing Letters* 110.21 (Oct. 2010), pp. 944–949. ISSN: 00200190. DOI: 10.1016/j.ipl.2010.07.024.

[31] Chromium.org. *Design Documents: RAPPOR (Randomized Aggregatable Privacy Preserving Ordinal Responses)*. URL: http://www.chromium.org/developers/design-documents/rappor (visited on 11/26/2018).

[32] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT press, 2009.

[33] Graham Cormode and S. Muthukrishnan. "An Improved Data Stream Summary: The Count-Min Sketch and Its Applications". In: *Journal of Algorithms* 55.1 (Apr. 2005), pp. 58–75. ISSN: 01966774. DOI: 10.1016/j.jalgor.2003.12.001.

[34] Graham Cormode, Magda Procopiuc, Divesh Srivastava, and Thanh T. L. Tran. "Differentially Private Publication of Sparse Data". In: (Mar. 4, 2011). arXiv: 1103.0825 [cs]. URL: http://arxiv.org/abs/1103.0825 (visited on 12/24/2018).

[35] Lawrence H Cox. "Network Models for Complementary Cell Suppression". In: *Journal of the American Statistical Association* 90.432 (1995), pp. 1453–1462.

[36] Lawrence H. Cox. "Suppression Methodology and Statistical Disclosure Control". In: *Journal of the American Statistical Association* 75 (1980), pp. 377–385.

[37] Bennett Cyphers. *Differential Privacy, Part 3: Extraordinary Claims Require Extraordinary Scrutiny*. Nov. 30, 2017. URL: https://www.accessnow.org/differential-privacy-part-3-extraordinary-claims-require-extraordinary-scrutiny/ (visited on 11/26/2018).

[38] Bennett Cyphers. *Understanding Differential Privacy and Why It Matters for Digital Rights*. Oct. 25, 2017. URL: https://www.accessnow.org/understanding-differential-privacy-matters-digital-rights/ (visited on 10/23/2018).

[39] Tore Dalenius. "A Simple Procedure for Controlled Rounding". In: *Statistik Tidskrift* 3 (1981), pp. 202–208.

[40] Tore Dalenius and Steven P. Reiss. "Data-Swapping: A Technique for Disclosure Control". In: *Journal of Statistical Planning and Inference* 6.1 (Jan. 1982), pp. 73–85. ISSN: 03783758. DOI: 10.1016/0378-3758(82)90058-1.

[41] *Dataflow Analysis & Differential Privacy for SQL Queries: Uber/Sql-Differential-Privacy*. Uber Open Source, Nov. 15, 2018. URL: https://github.com/uber/sql-differential-privacy (visited on 11/20/2018).

[42] Anindya De. "Lower Bounds in Differential Privacy". In: *Theory of Cryptography Conference*. Springer, Mar. 2012, pp. 321–338.

[43] Fan Deng and Davood Rafiei. "New Estimation Algorithms for Streaming Data: Count-Min Can Do More". In: (2007), p. 13.

[44] Dorothy E. Denning. "Secure Statistical Databases with Random Sample Queries". In: *ACM Transactions on Database Systems* 5.3 (Sept. 1, 1980), pp. 291–315. ISSN: 03625915. DOI: 10.1145/320613.320616.

[45] Dorothy E. Denning, Peter J. Denning, and Mayer D. Schwartz. "The Tracker: A Threat to Statistical Database Security". In: *ACM Transactions on Database Systems* 4.1 (Mar. 1, 1979), pp. 76–96. ISSN: 03625915. DOI: 10.1145/320064.320069.

[46] Dorothy Elizabeth Robling Denning. *Cryptography and Data Security*. Reading, Mass: Addison-Wesley, 1982. 400 pp. ISBN: 978-0-201-10150-8.

[47] Persi Diaconis and Bernd Sturmfels. "Algebraic Algorithms for Sampling from Conditional Distributions". In: *The Annals of Statistics* 26.1 (Feb. 1998), pp. 363–397. DOI: 10.1214/aos/1030563990.

[48] Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. "Detecting Violations of Differential Privacy". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security - CCS '18* (2018), pp. 475–489. DOI: 10.1145/3243734.3243818. arXiv: 1805.10277.

[49] Irit Dinur and Kobbi Nissim. "Revealing Information While Preserving Privacy". In: *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems - PODS '03*. The Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium. San Diego, California: ACM Press, 2003, pp. 202–210. ISBN: 978-1-58113-670-8. DOI: 10.1145/773153.773173.

[50] David Dobkin, Anita K. Jones, and Richard J. Lipton. "Secure Databases: Protection against User Influence". In: *ACM Transactions on Database Systems* 4.1 (Mar. 1, 1979), pp. 97–106. ISSN: 03625915. DOI: 10.1145/320064.320068.

[51] J. Domingo-Ferrer and J. Soria-Comas. "From T-Closeness to Differential Privacy and Vice Versa in Data Anonymization". In: *Knowledge-Based Systems* 74 (Jan. 2015), pp. 151–158. ISSN: 09507051. DOI: 10.1016/j.knosys.2014.11.011. arXiv: 1512.05110.

[52] Wenliang Du and Mikhail J Atallah. "A Study of Several Specific Secure Two-Party Computation Problems". In: *Purdue University, West Lafayette, IN* (2001). URL: http://www.cis.syr.edu/~wedu/Research/paper/duthesis.pdf (visited on 11/08/2018).

[53] Wenliang Du and Zhijun Zhan. "A Practical Approach to Solve Secure Multi-Party Computation Problems". In: *Proceedings of the 2002 workshop on New security paradigms* (2002), pp. 127–135.

[54] George T Duncan and Stephen E Fienberg. "Obtaining Information While Preserving Privacy: A Markov Perturbation Method for Tabular Data". In: (1997), p. 13.

[55] George T Duncan, Stephen E Fienberg, Ramayya Krishnan, Rema Padman, and Stephen F Roehrig. "Disclosure Limitation Methods and Information Loss for Tabular Data". In: *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies* (2001), pp. 135–166.

[56] Cynthia Dwork. "A Firm Foundation for Private Data Analysis". In: *Communications of the ACM* 54.1 (Jan. 1, 2011), p. 86. ISSN: 00010782. DOI: 10.1145/1866739.1866758.

[57] Cynthia Dwork. "An Ad Omnia Approach to Defining and Achieving Private Data Analysis". In: *Privacy, Security, and Trust in KDD*. Ed. by Francesco Bonchi, Elena Ferrari, Bradley Malin, and Yücel Saygin. Vol. 4890. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–13. ISBN: 978-3-540-78477-7 978-3-540-78478-4. DOI: 10.1007/978-3-540-78478-4_1. URL: http://link.springer.com/10.1007/978-3-540-78478-4_1 (visited on 11/07/2018).

[58] Cynthia Dwork. "Differential Privacy". In: *Automata, Languages and Programming*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener. Red. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, and Gerhard Weikum. Vol. 4052. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12. ISBN: 978-3-540-35907-4 978-3-540-35908-1. DOI: 10.1007/11787006_1. URL: http://link.springer.com/10.1007/11787006_1 (visited on 09/06/2018).

[59] Cynthia Dwork. "Differential Privacy: A Survey of Results". In: *Theory and Applications of Models of Computation*. Ed. by Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li. Vol. 4978. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–19. ISBN: 978-3-540-79227-7 978-3-540-79228-4. DOI: 10.1007/978-3-540-79228-4_38. URL: http://link.springer.com/10.1007/978-3-540-79228-4_38 (visited on 07/18/2018).

[60] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. "Our Data, Ourselves: Privacy Via Distributed Noise Generation". In: *Advances in Cryptology - EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 486–503. ISBN: 978-3-540-34546-6 978-3-540-34547-3. DOI: 10.1007/11761679_29. URL: http://link.springer.com/10.1007/11761679_29 (visited on 10/22/2018).

[61] Cynthia Dwork and Jing Lei. "Differential Privacy and Robust Statistics". In: ACM Press, 2009, p. 371. ISBN: 978-1-60558-506-2. DOI: 10.1145/1536414.1536466.

[62] Cynthia Dwork and Frank D. McSherry. "Differential Data Privacy". U.S. pat. 7698250B2. Microsoft Corp. Apr. 13, 2010. URL: https://patents.google.com/patent/US7698250B2/en (visited on 10/19/2018).

[63] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography*. Ed. by Shai Halevi and Tal Rabin. Red. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, and Gerhard Weikum. Vol. 3876. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284. ISBN: 978-3-540-32731-8 978-3-540-32732-5. DOI: 10.1007/11681878_14. URL: http://link.springer.com/10.1007/11681878_14 (visited on 09/06/2018).

[64] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Differential Privacy: A Primer for the Perplexed". In: Conference of European Statisticians. Tarragona, Spain, Oct. 2011, p. 8.

[65] Cynthia Dwork and Moni Naor. "On the Difficulties of Disclosure Prevention in Statistical Databases or The Case for Differential Privacy". In: *Journal of Privacy and Confidentiality* 2.1 (Sept. 1, 2010). ISSN: 2575-8527. DOI: 10.29012/jpc.v2i1.585.

[66] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: *Foundations and Trends® in Theoretical Computer Science* 9.3-4 (2013), pp. 211–407. ISSN: 1551-305X, 1551-3068. DOI: 10.1561/0400000042.

[67] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. "Boosting and Differential Privacy". In: IEEE, Oct. 2010, pp. 51–60. ISBN: 978-1-4244-8525-3. DOI: 10.1109/FOCS.2010.12.

[68] Cynthia Dwork and Adam Smith. "Differential Privacy for Statistics: What We Know and What We Want to Learn". In: *Journal of Privacy and Confidentiality* 1.2 (Apr. 1, 2010). ISSN: 2575-8527. DOI: 10.29012/jpc.v1i2.570.

[69] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response". In: ACM Press, 2014, pp. 1054–1067. ISBN: 978-1-4503-2957-6. DOI: 10.1145/2660267.2660348.

[70] Christos Faloutsos, H. V. Jagadish, and Nikolaos D. Sidiropoulos. "Recovering Information from Summary Data". In: (1997). URL: https://drum.lib.umd.edu/bitstream/handle/1903/5844/TR_97-7.pdf?sequence=1&isAllowed=y (visited on 11/08/2018).

[71] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. "Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries". In: (Mar. 3, 2015). arXiv: 1503.01214 [cs]. URL: http://arxiv.org/abs/1503.01214 (visited on 10/19/2018).

[72] Ivan P Fellegi. "On the Question of Statistical Confidentiality". In: *Journal of the American Statistical Association* 67.337 (1972), pp. 7–18.

[73] Sam Fletcher and Md Zahidul Islam. "Differentially Private Random Decision Forests Using Smooth Sensitivity". In: *Expert Systems with Applications* 78 (July 2017), pp. 16–31. ISSN: 09574174. DOI: 10.1016/j.eswa.2017.01.034.

[74] Arik Friedman and Assaf Schuster. "Data Mining with Differential Privacy". In: ACM Press, 2010, p. 493. ISBN: 978-1-4503-0055-1. DOI: 10.1145/1835804.1835868.

[75] *Functional Mechanism*. URL: https://sourceforge.net/projects/functionalmecha/ (visited on 12/26/2018).

[76] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Play ANY Mental Game or A Completeness Theorem for Protocols with Honest Majority." In: *In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC '87)*. New York, NY, USA: ACM, May 1987, pp. 218–229. DOI: https://doi.org/10.1145/28395.

[77] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. "Secure Multiparty Aggregation with Differential Privacy: A Comparative Study". In: *Proceedings of the Joint EDBT/ICDT 2013 Workshops* (2013), p. 9.

[78] Andy Greenberg. "How One of Apple's Key Privacy Safeguards Falls Short". In: *Wired* (Sept. 15, 2017). ISSN: 1059-1028. URL: https://www.wired.com/story/apple-differential-privacy-shortcomings/ (visited on 12/02/2018).

[79] Isabelle Guyon and André Elisseeff. "An Introduction to Variable and Feature Selection". In: *Journal of Machine Learning Research* 3 3 (Mar. 2003), pp. 1157–1182.

[80] Robert Hall, Larry Wasserman, and Alessandro Rinaldo. "Random Differential Privacy". In: *Journal of Privacy and Confidentiality* 4.2 (Mar. 1, 2013). ISSN: 2575-8527. DOI: 10.29012/jpc.v4i2.621.

[81] Moritz Hardt and Kunal Talwar. "On the Geometry of Differential Privacy". In: *Proceedings of the 42nd ACM Symposium on Theory of Computing - STOC '10*. The 42nd ACM Symposium. Cambridge, Massachusetts, USA: ACM Press, 2010, p. 705. ISBN: 978-1-4503-0050-6. DOI: 10.1145/1806689.1806786.

[82] David Harrison and Daniel L Rubinfeld. "Hedonic Housing Prices and the Demand for Clean Air". In: *Journal of Environmental Economics and Management* 5.1 (Mar. 1978), pp. 81–102. ISSN: 00950696. DOI: 10.1016/0095-0696(78)90006-2.

[83] Florian Hartmann. *One-off Notebook for Performing RAPPOR Aggregations: Mozilla/Rappor-Aggregator*. Mozilla, June 4, 2018. URL: https://github.com/mozilla/rappor-aggregator (visited on 11/26/2018).

[84] Xueyang Hu, Mingxuan Yuan, Jianguo Yao, Yu Deng, Lei Chen, Qiang Yang, Haibing Guan, and Jia Zeng. "Differential Privacy in Telco Big Data Platform". In: *Proceedings of the VLDB Endowment* 8.12 (Aug. 1, 2015), pp. 1692–1703. ISSN: 21508097. DOI: 10.14778/2824032.2824067.

[85] Ali Inan, Murat Kantarcioglu, Gabriel Ghinita, and Elisa Bertino. "Private Record Matching Using Differential Privacy". In: ACM Press, 2010, p. 123. ISBN: 978-1-60558-945-9. DOI: 10.1145/1739041.1739059.

[86] Vijay S Iyengar. "Transforming Data to Satisfy Privacy Constraints". In: (July 3, 2002), p. 10.

[87] Priyank Jain, Manasi Gyanchandani, and Nilay Khare. "Differential Privacy: Its Technological Prescriptive Using Big Data". In: *Journal of Big Data* 5.1 (Dec. 2018). ISSN: 2196-1115. DOI: 10.1186/s40537-018-0124-9.

[88] Zhanglong Ji, Zachary C. Lipton, and Charles Elkan. "Differential Privacy and Machine Learning: A Survey and Review". In: (Dec. 23, 2014). arXiv: 1412.7584 [cs]. URL: http://arxiv.org/abs/1412.7584 (visited on 07/18/2018).

[89] Noah Johnson, Joseph P Near, and Dawn Song. "Towards Practical Differential Privacy for SQL Queries". In: *Proceedings of the VLDB Endowment*. Vol. 526–539. 2018, p. 14.

[90] Oliver Kunz. "On the Equivalence Between One-Dimensional Discrete Walsh-Hadamard and Multidimensional Discrete Fourier Transforms". In: *IEEE Transactions on Computers* C-28.3 (Mar. 1979), pp. 267–268. ISSN: 0018-9340. DOI: 10.1109/TC.1979.1675334.

[91] Andrei Lapets, Frederick Jansen, Kinan Dak Albab, Rawane Issa, Lucy Qin, Mayank Varia, and Azer Bestavros. "Accessible Privacy-Preserving Web-Based Data Analysis for Assessing and Addressing Economic Inequalities". In: *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS) - COMPASS '18*. The 1st ACM SIGCAS Conference. Menlo Park and San Jose, CA, USA: ACM Press, 2018, pp. 1–5. ISBN: 978-1-4503-5816-3. DOI: 10.1145/3209811.3212701.

[92] Jaewoo Lee and Chris Clifton. "How Much Is Enough? Choosing for Differential Privacy". In: *International Conference on Information Security*. Springer, Berlin, Heidelberg, Oct. 26, 2011, pp. 325–340.

[93] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan. "Mondrian Multidimensional K-Anonymity". In: *22nd International Conference on Data Engineering (ICDE'06)*. 22nd International Conference on Data Engineering (ICDE'06). Atlanta, GA, USA: IEEE, 2006, pp. 25–25. ISBN: 978-0-7695-2570-9. DOI: 10.1109/ICDE.2006.101.

[94] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. "Incognito: Efficient Full-Domain K-Anonymity". In: *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data - SIGMOD '05*. The 2005 ACM SIGMOD International Conference. Baltimore, Maryland: ACM Press, 2005, p. 49. ISBN: 978-1-59593-060-6. DOI: 10.1145/1066157.1066164.

[95] Ezio Lefons. "An Analytic Approach to Statistical Databases". In: (1983), p. 15.

[96] Jing Lei. "Differentially Private M-Estimators". In: *Advances in Neural Information Processing Systems*. 2011, pp. 361–369.

[97] Hongwei Li, Yuanshun Dai, and Xiaodong Lin. "Efficient E-Health Data Release with Consistency Guarantee under Differential Privacy". In: *2015 17th International Conference on E-Health Networking, Application & Services (HealthCom)*. 2015 17th International Conference on E-Health Networking, Application & Services (HealthCom). Boston, MA, USA: IEEE, Oct. 2015, pp. 602–608. ISBN: 978-1-4673-8325-7. DOI: 10.1109/HealthCom.2015.7454576.

[98] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. "Closeness: A New Privacy Measure for Data Publishing". In: *IEEE Transactions on Knowledge and Data Engineering* 22.7 (July 2010), pp. 943–956. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.139.

[99] Ninghui Li, Tiancheng Li, Suresh Venkatasubramanian, and T Labs. "T-Closeness: Privacy Beyond k-Anonymity and -Diversity". In: *IEEE 23rd International Conference on Data Engineering 2007*. ICDE 2007. IEEE, 2007, pp. 106–115.

[100] Ninghui Li, Wahbeh Qardaji, and Dong Su. "On Sampling, Anonymization, and Differential Privacy: Or, k-Anonymization Meets Differential Privacy". In: (Jan. 13, 2011). arXiv: 1101.2604 [cs]. URL: http://arxiv.org/abs/1101.2604 (visited on 07/18/2018).

[101] Ninghui Li, Wahbeh Qardaji, Dong Su, and Jianneng Cao. "PrivBasis: Frequent Itemset Mining with Differential Privacy". In: (July 31, 2012). arXiv: 1208.0093 [cs]. URL: http://arxiv.org/abs/1208.0093 (visited on 07/18/2018).

[102] Chong K. Liew, Uinam J. Choi, and Chung J. Liew. "A Data Distortion by Probability Distribution". In: *ACM Transactions on Database Systems* 10.3 (Sept. 1, 1985), pp. 395–411. ISSN: 03625915. DOI: 10.1145/3979.4017.

[103] Yehuda Lindell. "Secure Multiparty Computation for Privacy-Preserving Data Mining". In: *Encyclopedia of Data Warehousing and Mining* (2005), pp. 1005–1009.

[104] Yehuda Lindell and Jonathan Katz. *Introduction to Modern Cryptography*. 2nd ed. Chapman and Hall/CRC, 2014.

[105] Yehuda Lindell and Eran Omri. "A Practical Application of Differential Privacy to Personalized Online Advertising". In: *IACR Cryptology EPrint Archive* 152 (Mar. 2011), p. 20.

[106] *Linnet Taylor - Group Privacy: Big Data and the Collective, MyData 2017*. URL: https://www.youtube.com/watch?v=BsZ05MVFXLU (visited on 01/03/2019).

[107] Chang Liu, Xiao Shaun Wang, Kartik Nayak, Yan Huang, and Elaine Shi. "ObliVM: A Programming Framework for Secure Computation". In: *2015 IEEE Symposium on Security and Privacy*. 2015 IEEE Symposium on Security and Privacy (SP). San Jose, CA: IEEE, May 2015, pp. 359–376. ISBN: 978-1-4673-6949-7. DOI: 10.1109/SP.2015.29.

[108] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. "Dependence Makes You Vulnerable: Differential Privacy Under Dependent Tuples". In: *Proceedings 2016 Network and Distributed System Security Symposium*. Network and Distributed System Security Symposium. San Diego, CA: Internet Society, 2016. ISBN: 978-1-891562-41-9. DOI: 10.14722/ndss.2016.23279.

[109] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. "L-Diversity: Privacy beyond k-Anonymity". In: *IEEE* (2006), p. 24. ISSN: 15564681. DOI: 10.1145/1217299.1217302.

[110] Norman S. Matloff. "Another Look at the Use of Noise Addition for Database Security". In: *1986 IEEE Symposium on Security and Privacy*. 1986 IEEE Symposium on Security and Privacy. Oakland, CA, USA: IEEE, Apr. 1986, pp. 173–173. ISBN: 978-0-8186-0716-5. DOI: 10.1109/SP.1986.10014.

[111] Andrew McGregor, Ilya Mironov, Toniann Pitassi, Omer Reingold, Kunal Talwar, and Salil Vadhan. "The Limits of Two-Party Differential Privacy". In: IEEE, Oct. 2010, pp. 81–90. ISBN: 978-1-4244-8525-3. DOI: 10.1109/FOCS.2010.14.

[112] Frank McSherry. "Privacy Integrated Queries". In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. ACM, 2009, pp. 19–30.

[113] Frank McSherry and Kunal Talwar. "Mechanism Design via Differential Privacy". In: *Foundations of Computer Science, 2007*. 48th Annual IEEE Symposium. IEEE, 2007, p. 10.

[114] Adam Meyerson and Ryan Williams. "On the Complexity of Optimal K-Anonymity". In: *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems - PODS '04*. The Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium. Paris, France: ACM Press, 2004, p. 223. ISBN: 978-1-58113-858-0. DOI: 10.1145/1055558.1055591.

[115] Gerome Miklau and Dan Suciu. "Controlling Access to Published Data Using Cryptography". In: *Proceedings 2003 VLDB Conference*. Elsevier, 2003, pp. 898–909. ISBN: 978-0-12-722442-8. DOI: 10.1016/B978-012722442-8/50084-7. URL: http://linkinghub.elsevier.com/retrieve/pii/B9780127224428500847 (visited on 11/08/2018).

[116] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. "Computational Differential Privacy". In: *Advances in Cryptology - CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 126–142. ISBN: 978-3-642-03355-1 978-3-642-03356-8. DOI: 10.1007/978-3-642-03356-8_8. URL: http://link.springer.com/10.1007/978-3-642-03356-8_8 (visited on 07/18/2018).

[117] Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. "GUPT: Privacy Preserving Data Analysis Made Easy". In: *Proceedings of the 2012 International Conference on Management of Data - SIGMOD '12*. The 2012 International Conference. Scottsdale, Arizona, USA: ACM Press, 2012, p. 349. ISBN: 978-1-4503-1247-9. DOI: 10.1145/2213836.2213876.

[118] Jack Murtagh and Salil Vadhan. "The Complexity of Computing the Optimal Composition of Differential Privacy". In: (July 11, 2015). arXiv: 1507.03113 [cs]. URL: http://arxiv.org/abs/1507.03113 (visited on 09/07/2018).

[119] Raymond H. Myers and Raymond H. Myers. *Classical and Modern Regression with Applications*. Vol. 2. Duxbury press Belmont, CA, 1990.

[120] Qun Ni and Alberto Trombetta. "Privacy-Aware Role Based Access Control". In: *ACM Transactions on Information and System Security (TISSEC)* 13.3 (2010), p. 24.

[121] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (Oct 2011), pp. 2825–2830. URL: http://www.jmlr.org/papers/v12/pedregosa11a.html (visited on 01/06/2017).

[122] Mor Peleg, Dizza Beimel, Dov Dori, and Yaron Denekamp. "Situation-Based Access Control: Privacy Management via Modeling of Patient Data Access Scenarios". In: *Journal of Biomedical Informatics* 41.6 (Dec. 2008), pp. 1028–1040. ISSN: 15320464. DOI: 10.1016/j.jbi.2008.03.014.

[123] J. R. Quinlan. "Induction of Decision Trees". In: *Machine Learning* 1.1 (Mar. 1986), pp. 81–106. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00116251.

[124] *RAPPOR: Privacy-Preserving Reporting Algorithms. Contribute to Google/Rappor Development by Creating an Account on GitHub*. Google, Nov. 19, 2018. URL: https://github.com/google/rappor (visited on 11/20/2018).

[125] A. Rastogi, M. A. Hammer, and M. Hicks. "Wysteria: A Programming Language for Generic, Mixed-Mode Multiparty Computations". In: *2014 IEEE Symposium on Security and Privacy*. 2014 IEEE Symposium on Security and Privacy. May 2014, pp. 655–670. DOI: 10.1109/SP.2014.48.

[126] Vibhor Rastogi and Suman Nath. "Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption". In: *Proceedings of the 2010 International Conference on Management of Data - SIGMOD '10*. The 2010 International Conference. Indianapolis, Indiana, USA: ACM Press, 2010, p. 735. ISBN: 978-1-4503-0032-2. DOI: 10.1145/1807167.1807247.

[127] Steven P Reiss. "Practical Data-Swapping: The First Steps". In: *ACM Transactions on Database Systems* 9.1 (Mar. 1984), pp. 20–37.

[128] Roberto. *Differential Privacy for Dummies*. July 29, 2016. URL: https://robertovitillo.com/2016/07/29/differential-privacy-for-dummies/ (visited on 10/19/2018).

[129] Lior Rokach and Oded Maimon. "Decision Trees". In: *Data Mining and Knowledge Discovery Handbook*. Springer, 2005, pp. 165–192. URL: http://link.springer.com/chapter/10.1007/0-387-25465-X_9 (visited on 02/17/2017).

[130] Indrajit Roy, Hany Ramadan, Srinath Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. "Airavat: Security and Privacy for MapReduce". In: *NSDI* 10 (2010), pp. 297–312.

[131] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. "The Earth Mover's Distance as a Metric for Image Retrieval". In: (2000), p. 23.

[132] P. Samarati. "Protecting Respondents Identities in Microdata Release". In: *IEEE Transactions on Knowledge and Data Engineering* 13.6 (Nov.-Dec./2001), pp. 1010–1027. ISSN: 10414347. DOI: 10.1109/69.971193.

[133] Pierangela Samarati and Latanya Sweeney. "Protecting Privacy When Disclosing Information: K-Anonymity and Its Enforcement through Generalization and Suppression". In: (1998), p. 19.

[134] Rathindra Sarathy and Krishnamurty Muralidhar. "Evaluating Laplace Noise Addition to Satisfy Differential Privacy for Numeric Data". In: *Trans. Data Privacy* 4.1 (2011), pp. 1–17. URL: https://www.researchgate.net/profile/Rathindra_Sarathy/publication/220095234_Evaluating_Laplace_Noise_Addition_to_Satisfy_Differential_Privacy_for_Numeric_Data/links/00b7d5294fe279c7e9000000/Evaluating-Laplace-Noise-Addition-to-Satisfy-Differential-Privacy-for-Numeric-Data.pdf (visited on 11/07/2018).

[135] Jan Schlörer. "Identification and Retrieval of Personal Records from a Statistical Data Bank". In: *Methods of information in medicine* 14.01 (1975), pp. 7–13.

[136] Adi Shamir. "How to Share a Secret". In: *Communications of the ACM* 22.11 (1979), pp. 612–613. URL: http://www.dtic.mil/dtic/tr/fulltext/u2/a069397.pdf (visited on 11/09/2018).

[137] Reza Shokri and Vitaly Shmatikov. "Privacy-Preserving Deep Learning". In: ACM Press, 2015, pp. 1310–1321. ISBN: 978-1-4503-3832-5. DOI: 10.1145/2810103.2813687.

[138] Arie Shoshani. "Statistical Databases: Characteristics, Problems, and Some Solutions." In: *VLDB* 82 (1982), pp. 208–222.

[139] Adam Smith. "Privacy-Preserving Statistical Estimation with Optimal Convergence Rates". In: *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing - STOC '11*. The 43rd Annual ACM Symposium. San Jose, California, USA: ACM Press, 2011, p. 813. ISBN: 978-1-4503-0691-1. DOI: 10.1145/1993636.1993743.

[140] Adam Smith and Abhradeep Thakurta. "Differentially Private Model Selection via Stability Arguments and the Robustness of the Lasso". In: *Conference on Learning Theory*. 2013, pp. 819–850.

[141] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, and Sergio Martínez. "Enhancing Data Utility in Differential Privacy via Microaggregation-Based $$k$$ k -Anonymity". In: *The VLDB Journal* 23.5 (Oct. 2014), pp. 771–794. ISSN: 1066-8888, 0949-877X. DOI: 10.1007/s00778-014-0351-4.

[142] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, and Sergio Martínez. "T-Closeness through Microaggregation: Strict Privacy with Enhanced Utility Preservation". In: *IEEE Transactions on Knowledge and Data Engineering* 27.11 (Nov. 1, 2015), pp. 3098–3110. ISSN: 1041-4347. DOI: 10.1109/TKDE.2015.2435777. arXiv: 1512.02909.

[143] Jordi Soria-Comas and Josep Domingo-Ferrert. "Differential Privacy via T-Closeness in Data Publishing". In: *2013 Eleventh Annual Conference on Privacy, Security and Trust*. 2013 Eleventh Annual Conference on Privacy, Security and Trust (PST). Tarragona, Spain: IEEE, July 2013, pp. 27–35. ISBN: 978-1-4673-5839-2. DOI: 10.1109/PST.2013.6596033.

[144] Thomas Steinke and Jonathan Ullman. "Between Pure and Approximate Differential Privacy". In: *Journal of Privacy and Confidentiality* 7.2 (Jan. 12, 2017). ISSN: 2575-8527. DOI: 10.29012/jpc.v7i2.648.

[145] Shanmuga Sundari and M Ananthi. "Secure Multi-Party Computation in Differential Private Data with Data Integrity Protection". In: *2015 International Conference on Computing and Communications Technologies (ICCCT)*. 2015 International Conference on Computing and Communications Technologies (ICCCT). Chennai, India: IEEE, Feb. 2015, pp. 180–184. ISBN: 978-1-4799-7623-2. DOI: 10.1109/ICCCT2.2015.7292742.

[146] Latanya Sweeney. "ACHIEVING K-ANONYMITY PRIVACY PROTECTION USING GENERALIZATION AND SUPPRESSION". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (Oct. 2002), pp. 571–588. ISSN: 0218-4885, 1793-6411. DOI: 10.1142/S021848850200165X.

[147] Latanya Sweeney. "K-Anonymity: A Model for Protecting Privacy". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (Oct. 2002), pp. 557–570. ISSN: 0218-4885, 1793-6411. DOI: 10.1142/S0218488502001648.

[148] Latanya Sweeney. "Weaving Technology and Policy Together to Maintain Confidentiality". In: *The Journal of Law, Medicine & Ethics* 25.2-3 (June 1997), pp. 98–110. ISSN: 1073-1105, 1748-720X. DOI: 10.1111/j.1748-720X.1997.tb01885.x.

[149] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. "Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12". In: (Sept. 8, 2017). arXiv: 1709.02753 [cs]. URL: http://arxiv.org/abs/1709.02753 (visited on 07/18/2018).

[150] Linnet Taylor and Luciano Floridi. "Group Privacy: New Challenges of Data Technologies". In: *Group Privacy* (2017), p. 293.

[151] Abhradeep Guha Thakurta, Andrew H. Vyrros, Umesh S. Vaishampayan, Gaurav Kapoor, Julien Freudiger, Vivek Rangarajan Sridhar, and Doug Davidson. "Learning New Words". U.S. pat. 9645998B1. Apple Inc. May 9, 2017. URL: https://patents.google.com/patent/US9645998/en (visited on 12/05/2018).

[152] Abhradeep Guha Thakurta, Andrew H. Vyrros, Umesh S. Vaishampayan, Gaurav Kapoor, Julien Freudiger, Vivek Rangarajan Sridhar, and Doug Davidson. "Learning New Words". U.S. pat. 9594741B1. Apple Inc. Mar. 14, 2017. URL: https://patents.google.com/patent/US9594741/en (visited on 12/05/2018).

[153] Abhradeep Guha Thakurta, Andrew H. Vyrros, Umesh S. Vaishampayan, Gaurav Kapoor, Julien Freudinger, Vipul Ved Prakash, Arnaud Legendre, and Steven Duplinsky. "Emoji Frequency Detection and Deep Link Frequency". U.S. pat. 9705908B1. Apple Inc. July 11, 2017. URL: https://patents.google.com/patent/US9705908/en (visited on 12/05/2018).

[154] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. URL: https://www.jstor.org/stable/2346178 (visited on 11/20/2018).

[155] J. F. Traub, Y. Yemini, and H. Woźniakowski. "The Statistical Security of a Statistical Database". In: *ACM Transactions on Database Systems* 9.4 (Nov. 5, 1984), pp. 672–679. ISSN: 03625915. DOI: 10.1145/1994.383392.

[156] T.M. Truta and B. Vinay. "Privacy Protection: P-Sensitive k-Anonymity Property". In: *22nd International Conference on Data Engineering Workshops (ICDEW'06)*. 22nd International Conference on Data Engineering Workshops (ICDEW'06). Atlanta, GA, USA: IEEE, 2006, pp. 94–94. ISBN: 978-0-7695-2571-6. DOI: 10.1109/ICDEW.2006.116.

[157] Stanley L Warner. "Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias". In: *Journal of the American Statistical Association* 60.309 (1965), pp. 63–69.

[158] Leon Willenborg and Ton de Waal. *Statistical Disclosure Control in Practice*. Springer Science & Business Media, Apr. 11, 1996. 172 pp. ISBN: 978-0-387-94722-8.

[159] Rick L Wilson and Peter A Rosen. "Protecting Data through 'Perturbation' Techniques: The Impact on Knowledge Discovery in Databases". In: (2003), p. 13.

[160] Xiaokui Xiao and Yufei Tao. "Personalized Privacy Preservation". In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data - SIGMOD '06*. The 2006 ACM SIGMOD International Conference. Chicago, IL, USA: ACM Press, 2006, p. 229. ISBN: 978-1-59593-434-5. DOI: 10.1145/1142473.1142500.

[161] Xiaochun Yang and Chen Li. "Secure XML Publishing without Information Leakage in the Presence of Data Inference". In: *Proceedings of the Thirtieth International Conference on Very Large Data Bases*. Vol. 30. VLDB Endowment, 2004, pp. 96–107. URL: http://www.vldb.org/conf/2004/RS3P2.PDF (visited on 11/08/2018).

[162] Yin Yang, Zhenjie Zhang, Gerome Miklau, Marianne Winslett, and Xiaokui Xiao. "Differential Privacy in Data Publication and Analysis". In: *Proceedings of the 2012 International Conference on Management of Data - SIGMOD '12*. The 2012 International Conference. Scottsdale, Arizona, USA: ACM Press, 2012, p. 601. ISBN: 978-1-4503-1247-9. DOI: 10.1145/2213836.2213910.

[163] Andrew C Yao. "Protocols for Secure Computations". In: *3rd Annual Symposium on Foundations of Computer Science*. IEEE, 1982, pp. 160–164.

[164] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets". In: *27th Annual Symposium on Foundations of Computer Science, 1986*. Toronto, ON, Canada, Canada: IEEE, Oct. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.

[165] Clement Yu and Francis Chin. "A Study on the Protection of Statistical Data Bases". In: *Proceedings of the 1977 ACM SIGMOD international conference on Management of data* (Aug. 1997), pp. 169–181.

[166] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. "Functional Mechanism: Regression Analysis under Differential Privacy". In: (Aug. 1, 2012). arXiv: 1208.0219 [cs]. URL: http://arxiv.org/abs/1208.0219 (visited on 12/23/2018).

[167] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. "Privacy-Enhancing k-Anonymization of Customer Data". In: *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems - PODS '05*. The Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium. Baltimore, Maryland: ACM Press, 2005, p. 139. ISBN: 978-1-59593-062-0. DOI: 10.1145/1065167.1065185.

[168] Author Nina Zumel. *A Simpler Explanation of Differential Privacy*. Oct. 2, 2015. URL: http://www.win-vector.com/blog/2015/10/a-simpler-explanation-of-differential-privacy/ (visited on 10/18/2018).