The Strengths, Weaknesses and Promise of Differential Privacy as a Privacy-Protection Framework

Catherine Wright
Dept. of Computer Science
University of New Mexico
wrightc@unm.edu

Kellin Rumsey
Dept. of Mathematics and Statistics
University of New Mexico
knrumsey@unm.edu

***Abstract*—Protection of personal information inside a database is an important and challenging problems. With the advancements in theory of hardware over the past several decades, database systems are more than ever before vulnerable to database reconstruction and linkage attacks. We discuss Differential Privacy as a framework for privacy protection, which makes mathematical guarantees to individuals that they not be harmed by their participation in a database. We discuss query response methods which achieve differential privacy, and thoroughly examine the associated challenges.**

***Index Terms*—database reconstruction, differential privacy, linkage attacks**

I. Introduction

Statistical databases are an essential part of life in the 21^{st} century and lead directly to the betterment of society as a whole. These datasets are used daily to inform policy decisions, advise allocation of resources, lead to innovations in technology and are critical to scientific learning in fields such as medicine. Although statistical databases benefit society as a whole, recent hardware and theoretical advancements have made individuals vulnerable to adversarial privacy attacks. Although there is little or no consensus as to the "how", most agree that database distributers have an ethical obligation to protect the data of an individual to a reasonable extent.

Sensitive data publishers have long been aware of privacy concerns, and take elementary steps towards protection of each individuals privacy. Historically, this means withholding data at the individual level and publication of data only at the higher sub-group level. Even if data is published at high levels of aggregation, the original microdata can be reconstructed in what is known as a database reconstruction attack [1]. This reconstruction of microdata involves (i) specification of mathematical constraints, (ii) translation of constraints to boolean variables and (iii) solving the satisfiability (SAT) problem. The first step, while challenging in it's own ways, is not a computationally challenging task. Translation of these constraints can now be accomplished using special languages such as Sugar. Once the constraints have been translated, the SAT problem must be solved. Although the SAT problem is NP-complete, rapid progress over the last 2-3 decades has produced SAT solvers which use a variety of advanced heuristics to prune vast amounts of the search space and are capable of solving systems with over a million variables in just a couple of minutes [2]. As capabilities for parallel processing continue to improve, a well-equipped adversary can completely undermine the privacy of a vulnerable dataset.

There are at least three modern approaches for defense against adversarial reconstruction attacks, which we outline here. The first and most straightforward solution is to publish less data. This defensive strategy has the obvious drawback of limiting inferences which can be made about the population as a whole, limiting the usefulness of the database. If the theoretical promises of quantum computing for NP-hard problems ever become a reality or other unforeseen speed-ups lead to SAT solvers which continue to improve, even this seemingly obvious solution may become unreasonable. A related challenge is the determination of how much (or little) data is safe to publish. The second and third strategy are both related to noise injection, either to query results (output) or to the published data itself (input). Both of these ideas, along with a comprehensive comparison, will be discussed in greater detail throughout this paper. It is hopefully clear that such a method must depend on the noise injection level, with two extreme cases on either end of the spectrum. Adding too little noise to the data clearly compromises privacy of the data, but adding too much noise renders the dataset practically ineffectual for inferential purposes. To achieve a meaningful balance between these two extremes, a rigorous mathematical framework will be desired [3]. Although it comes equipped with its own set of challenges and weaknesses, differential privacy has shown a tremendous amount of promise in the search for such a framework.

The rest of this paper is outlined as follows. In section 2, we provide a general discussion of privacy requirements for private databases and some of the general challenges. Section 3 defines differential privacy and illustrates the theoretical application with a number of examples and Section 4 illustrates the extension of the framework to a more complicated setting. Section 5 provides a detailed discussion of the weaknesses of the differential privacy framework as the relate to practical implementations, and section 5 describes some modern attempts to overcome these practical challenges. Section 6 discusses the closely related notion of Pan-privacy and concluding remarks are found in section 7.

II. Private Databases

A. The "Why?" of Database Privacy

It is almost universally accepted that a database publisher should be required to keep anonymity and protect the privacy of its participants. We first note that not all datasets are created equally, and it is worth discussing what differences, if any, apply ethically to the publishers of these datasets. Data is collected primarily in three different ways: compulsory, altruistically and opportunistically [1]. Compulsory data, such as census or tax data, is often mandated by an official organization such as State or Country. Altruistic data refers to data which is willingly given by respondents who choose to share their data in hope that it will help inform decision making. This data may include medical or political survey responses. Opportunistic data refers to data which is obtained, often unknowingly to the participant, via circumstantial methods. This class of data, for instance, refers to Google search history, Snapchat selfie data, credit card history and twitter scraping.

Compulsory data, especially when compelled by governmental institutions, is required by law to abide by certain privacy requirements. In a similar vein, altruistic data donors rightfully expect that their personal data will be kept private. If altruistic data regularly violates this inherent agreement, there will quickly be few individuals who are willing to provide data. Opportunistic data is an interesting case, especially given it's relatively new emergence. These organizations are often privately owned or at least not held to the same legal requirements as compulsory data. There have been a number of privacy violations as of late, such as Facebook exposing the personal information of nearly 50 million users after a computer network attack in 2018 or Amazon sending 1,700 personal "Alexa" recordings to the wrong person [4]. If the public blowback alone is not enough to encourage these companies to protect the privacy of its users, ethical and legal requirements should. It seems reasonable that for the betterment of society, all forms of data should be held to (at least) a minimal requirement of privacy.

B. The "How?" of Database Privacy

Before we discuss the rigorous framework provided by differential privacy, we begin our discussion with a number of informal ideas for privacy protection of statistical databases [5]. A common suggestion is to restrict queries which ask about individuals or small groups and limit only to larger subgroups. Our previous discussion on microdata reconstruction through a database reconstruction attack should make it obvious that this approach will fall flat, but we give a simple example anyways. Suppose an adversary makes the following two queries about the entire dataset.

Q1: How many people in the database have blue eyes?

Q2: How many people in the database, not named Alan Turing, have blue eyes?

If these queries return different counts (clearly separated by one value), we have deduced that Alan Turing must have blue eyes. A simple fix, it would seem, is to disallow queries about unique identifiers such as name. Equipped with the mathematical and computational tools such as a fast SAT solver and the appropriate set of constraints however, a database reconstruction attack can be performed regardless of this attempt. If we add *auxiliary information* into the equation then matters get even worse, but more on this later. A natural follow up is the idea referred to as *query auditing*, which designates that each query should be evaluated in the context of the query history to determine if a query might disclose personal information. Such an approach however is a computational impossibility for large databases and query histories. In addition, we consider the blue-eyed Alan Turing example from above, and note that if the query system had chosen not to respond, this fact alone may be disclosive of Alan's eye color. Next, we recall the idea of publishing less data (discussed in the introduction) and suggest a related approach. Subsampling methods revolve around the idea of publishing only a sample of the population, denoted by a random sample of rows in the table. In some sense, this seems to achieve the balance between privacy and efficacy that we so desperately need. If the subsample is large, then the data will be useful for inference. If the population is larger still, so that the subsample is small in comparison, each persons privacy will be protected *on average*. Although "average privacy protection" sounds nice in principle, it means nothing to the unlucky individual whose social security number was just leaked. Subsampling methods can be improved, by applying them at the query level so that certain statistical inferences can be made for the population while having some level of probabilistic control over the privacy protection of an individual. If the consequences are high however, or if the population is small/moderate in size, this method too will fall far short and an individual will be harmed.

C. The "What?" of Database Privacy

This last example leads our discussion to an interesting arena. In the preceding paragraph, we discussed the shortcomings of several informal "hows" of private data analysis. Perhaps a more important question, for now at least, is "what"? In 1977, a statistician named Tore Dalenius proposed a privacy goal for statistical databases which was considered to be a gold standard for many years [**?**]. The idea is simple,

"Anything that can be learned about a respondent from the statistical database should be learnable without access to the database." - Tore Dalenius

Five years later, the concept of *Semantic security* was introduced to the world, and is strikingly similar to Dalenius' goal. Even today, Semantic security is considered the gold standard for security of cryptosystems. Semantic security states that the encryption of a message should reveal nothing about the message itself. In fact, the lack of semantic security was precisely the ingredient needed for the real Alan Turing (et. al.) to crack Enigma during WWII. Semantic security is formalized by requiring an optimal adversary with exhaustive auxiliary information to gain no information when he/she obtains access to the ciphertext. Attempts to formalize Dalenius' criterion for private databases usually boil down to Bayesian approaches, in which a characteristic or feature of an individual is assigned a prior distribution. With each query, the prior distribution is "updated" and turned into a posterior distribution. If Dalenius' goal is to be achieved, we should require that the prior and posterior distributions are "not too different", measured perhaps by Kulback Leibler divergence or some similar metric [8]. This idea, which works so well in related fields, feels intuitively like the right direction. It feels like the benchmark criterion we have been looking for. So what goes wrong? Let us turn again to our imaginary blue eyed Alan Turing. Suppose Alan Turing proudly tweets that his eye color is the third most common eye color in all of England. A strange thing to post to Twitter, but let us carry on with the hypothetical for illustrations sake. Alan Turing has revealed nothing directly about his eye color, but if a quick database query reveals that the third most common eye color in Britain is "blue", we have learned something "private" about Alan Turing that we did not know before. This use of *auxiliary information* leads to what we call a *linkage attack*. This silly hypothetical example can be formalized by a rigorous impossibility result, and the entirety of Dalenius' desideratum comes crashing down. Turing did not even need to be a member of the database to be "hurt" by this adversarial attack. Surely we can agree however, that it is not the publishers of this eye-color database responsibility to protect Alan Turing's private information from this sort of an attack.

Whether the data is compulsory, opportunistic or altruistic, Dalenius' goal is unreasonable in far too many settings, especially as hybrid opportunistic/altruistic data becomes more readily available in the modern era of social media and big data. Thus we need to seek a more realistic goal. Publication of a database can by no means guarantee that no harm comes to an individual. Take another simple example for illustration. Consider a medical database which conclusively demonstrates that smoking severely increases the risk of cancer. There is no way to guarantee that a current smoker will not be harmed by this result. His insurance costs may increase and the state he lives in may raise tobacco taxes. Most would agree however, that this not the fault or responsibility of the agency who conducted the study or the agency who published/hosted the database. We certainly know more about the smoker as an individual after the results of the study than we did before. The key takeaway here, is that the impact on the smoker had nothing to do with his inclusion or exclusion from the study. The information gained on this individual, via auxiliary information, could have been gained regardless of his participation. It appears that a reasonable and achievable standard has just made itself known [7].

"[This form of] privacy promises to protect individuals from any **additional** *harm that they might face due to their data being in the private database x that they would not have faced had their data not been part of x."*

While it may not be as illustrious as Dalenius' unachievable goal, many have argued that this is in many ways sufficient. To the altruistic donor, this is more than sufficient for the perceived pros to outweigh the cons. A compulsory donor may still have reason to complain, but if the database is truly necessary then this requirement should be satisfactory. It turns out that this is exactly the promise made by differential privacy and, as we will shortly see, it takes a mathematically rigorous path in doing so.

III. Differential Privacy

The primary goal of differential privacy is to formalize the promise of the previous section. Put another way, differential privacy guarantees that the ability of an adversary to learn information about an individual is *almost* the same, independent of the individuals participation status in the dataset. To put this in the Bayesian framework, as we did for Dalenius, the distance between our prior and posterior distribution for an individual should be essentially independent of his inclusion or exclusion in the dataset. We formalize this notion by first considering pairs of datasets A and B which differ by exactly one row. This single row corresponds to an individual who is either being added or removed from the database. Returning to the idea of noise injection which was discussed in Section I, we consider a randomized perturbation function $\mathcal{K}$, which maps a dataset D to a possibly altered dataset $\mathcal{K}(D) = D'$. We are now ready to formally state the definition of differential privacy.

Definition: *A randomized function $\mathcal{K}$ provides ϵ-differential privacy if for all datasets A and B differing by exactly one row, we have*

$$P(\mathcal{K}(A) \in S) \leq e^{\epsilon} P(\mathcal{K}(B) \in S)$$

where S is an arbitrary subset of $Range(\mathcal{K})$.

To better understand this definition, let us fix our attention on a particular pair of datasets A which is complete and B which is identical to A with the omission of exactly one individual. Note that the definition given above applies in both directions. This yields the inequalities

$$P(\mathcal{K}(B) \in S)e^{-\epsilon} \leq P(\mathcal{K}(D) \in S) \leq P(\mathcal{K}(B) \in S)e^{\epsilon}.$$

The region defined by these inequalities in shown in Figure 1, where $P(\mathcal{K}(A) \in S)$ is taken (arbitrarily) to be 0.25. Notice that as $\epsilon \to 0$ we are requiring that the privacy mechanism yields $\mathcal{K}(B) \in S$ with exactly the same probability as $\mathcal{K}(A) \in S$ regardless of the single omitted individuals participation. As ϵ increases, this probability bound increases as well decreasing the strength of the guarantee.

To understand a 0-differentially private mechanism, consider the ridiculous (and completely useless) perturbation function

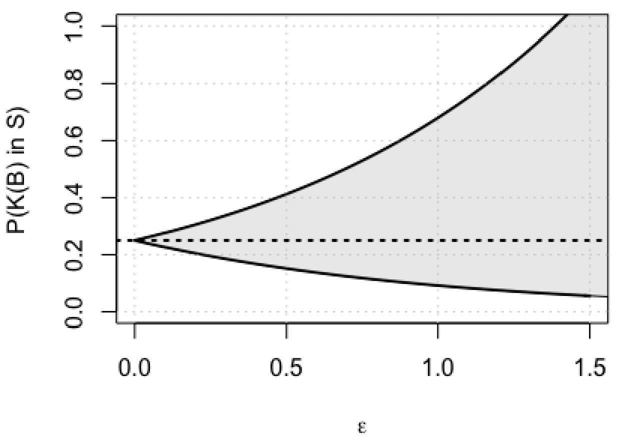

Fig. 1. Implications of ϵ-differential privacy. The horizontal dotted line represents $P(\mathcal{K}(A) \in S)$. The shaded region represents the ϵ-differential privacy equations for $P(\mathcal{K}(B) \in S)$

$\mathcal{K}_0$ which maps *any* dataset D to a dataset Z which consists entirely of random entries according to some fixed probability distribution. Clearly, this non-sensical privacy mechanism is outputting garbage which will provide no inference at all about the population of interest. An individuals information will always be protected trivially, and inclusion or exclusion from the dataset makes no difference at all. On the other extreme consider a not-so-random mechanism $\mathcal{K}_1$ which always maps D to itself (the identity function). There will always be datasets and queries which violate differential privacy for any reasonable ϵ value. To see this, we return to the example of section 2 where we first discovered that Alan Turing has blue eyes. Let S be the set of possible databases which such that the result of $Q1$ minus the result of $Q2$ is equal to 1. If A is the dataset with Alan and B is the dataset without him, then we have:

$$P(\mathcal{K}_1(A) \in S) = 1 \quad \text{and} \quad P(\mathcal{K}_1(B) \in S) = 0$$

It can immediately be seen that differential privacy cannot be achieved for any value of $\epsilon > 0$ no matter how large. Now that we have explored, somewhat humorously, these two extremes, we turn our attention to more realistic choices of $\mathcal{K}$.

A. *Achieving Differential Privacy*

As a first attempt at achieving differential privacy, we can utilize a rather old idea from the Statistics literature. Consider the following problem for a moment. The year is 1965 and researchers are interested in determining the proportion of High school students who use illegal drugs [9]. Because students will often lie about such a question, for fear of repercussion, the following scheme was used. The researcher asks the student to flip a coin. After the flip, the student is asked to answer yes or no to one of the following questions:

- If heads: Have you used an illegal drug in the last month?
- If tails: Was your birthday in the month of June?

Since the probability of the coin landing heads is known (1/2) and the probability of a birthday in June is also known (1/12), the researchers were able to adjust the inference on the unknown parameter of interest accordingly while protecting the privacy of the students. This scheme can be generalized for a larger class of problems and can be shown to be differentially private. Consider a property-count query which asks for the number of individuals in a dataset satisfying some property Q. For each individual in the dataset have them ”respond truthfully” (i.e. count them) with probability p and have them ”lie” with probability $1-p$. The *odds* of telling the truth is the ratio $p/(1-p)$, and it can be shown that this sampling scheme satisfies ϵ-differential privacy when ϵ is equal to the ”log odds” ($\epsilon = \ln(p) - \ln(1-p)$). We can analyze the inferential error using standard properties of the Binomial distribution. Let $b_i = 1$ if the i^{th} person in the database satisfies property Q and let C_i be a random variable (representing the coin flip) which is equal to 1 with probability p and 0 otherwise. Then the ”response” of the i^{th} individual is a random variable X_i defined by

$$X_i = b_iC_i + (1-b_i)(1-C_i)$$

Now the error in the response can be defined as $\Delta = \sum_{i=1}^n b_i - \sum_{i=1}^n X_i$. If the true answer to the query is m, we have the following results for the expected value and variance of Δ.

$$E(\Delta) = 2m(p-1) + n(1-p)$$
$$V(\Delta) = 4mp(1-p) + np(1-p)$$

We note that when $p \approx 1$ the error term is approximately zero. Unfortunately, if $p \in (0,1)$, a necessary requirement for protection of privacy, then the variance of the error term grows linearly with the sample size n. Let us summarize by considering two simple cases with $n = 1000$ and $m = 400$. Suppose the query ”lies” with probability $1/3$ leading to a $\ln(2)$ ($\epsilon = 0.69$) differentially private algorithm. The error term has an expected magnitude of 100 with a variance of 467 leading to nearly useless inference. We can decrease $1-p$ to $1/10$ shrinking the expected value of error to 30 and the variance to 189, but the algorithm is far less private with $\epsilon =$ 2.2. This approach is the one outlined in [7], and we note that the ”bias” of the response can be reduced via a careful bias adjustment, but the variance of the error always scales linearly with n. Although this is a decent first attempt at achieving differential privacy, we will see that we can do much better by achieving error on this type of query whose growth is *constant* as a function of n.

B. *Improving Differential Privacy with Laplacian Noise*

In order to achieve differential privacy, we must build around the idea of hiding the participation of a single individual. We start by examining a few simple queries in particular for a hypothetical database containing salary information for some population.

- How many individuals in the database have a salary of more than $\$45,000$?
- What is the average salary of individuals in the database?
- What is the maximum salary of individuals in the database?

The first query is ideal for differential privacy, since participation (or lack thereof) of an individual can change the answer by at most 1. If the dataset is relatively small and contains an outlier (i.e. the CEO of a medium size company), the second query can be strongly influenced by the inclusion or exclusion of this outlying individual. Nonetheless in many large data settings, or if skew and outliers are limited, this query might be easily handled by differential privacy. On the other hand, the third query presents a challenge since the maximum depends solely on a single individuals private data. If this individual opts out of the study and the next highest salary is significantly lower a tremendous amount of noise will need to be added to ensure differential privacy. We will discuss these practical shortcomings of differential privacy in more detail in Section 5. For now, we focus on queries of the first type, generally stated as ”how many rows in a database satisfy property Q”? Differential privacy for these types of queries can be ensured by adding noise to the query response in an appropriate manner. Consider two databases $A \subset B$ differing by a single row, and let S be the set which contains all databases whose answer to the property-count query lies between $[r-1, r+1]$. Suppose the true response on dataset B is r and the true response on dataset A is $r-1$. For differential privacy to hold, we need the privacy mechanism to add noise δ such that

$$P(r+\delta-1 \in [r-c, r+c]) \leq e^{\epsilon} P(r+\delta \in [r-c, r+c])$$

By standard inequality manipulation, the r drops out of this equation and rearranging yields:

$$\frac{P(1-c \leq \delta \leq c+1)}{P(-c \leq \delta \leq c)} \leq e^{\epsilon}$$

Thus when you shift the interval by at most 1 unit, the ratio of probabilities should be bounded above by e^{ϵ}. An identical argument (swapping the role of A and B in the definition) gives a lower bound of $e^{-\epsilon}$. So to satisfy ϵ-differential privacy for the property-count query, we only need to determine a distribution which has this property. This property can be easily guaranteed by using noise δ which is distributed according to a Laplacian (or double exponential) distribution with density function

$$p(z|\epsilon) = \frac{\epsilon}{2} \exp(-\epsilon|z|), \ \ z \in \mathbb{R}$$

The variance of the Laplace distribution is $2/\epsilon^2$ and it is centered at 0. Consider the first query in our hypothetical salary database where B is the complete data and A is identical except for the removal of the CEO. The true answer to this query with respect to B is 205 and is 204 with respect to A. Figures 2 and 3 illustrate the trade-off between precise inference and privacy protection as ϵ decreases from 1 to 0.2. When ϵ takes on larger values such as 1, we maintain the ability to perform precise inference on the population of interest at a privacy cost. Decreasing ϵ strengthens the promise of differential privacy, but inference necessarily becomes less precise.

One particularly nice feature of this approach, is that the variance of the error is constant, independent of the number of dataset rows or sample size n. If the true response to a property-count query is m, then the response to the query is $m+\delta$, $\delta \sim Laplace(\epsilon)$ so that the error is 0 on average and has a variance of $2/\epsilon^2$ which is independent of n. In other words, the error of this approach is $\mathcal{O}(\epsilon)$ compared to the $\mathcal{O}(\sqrt{n}e^{\epsilon})$

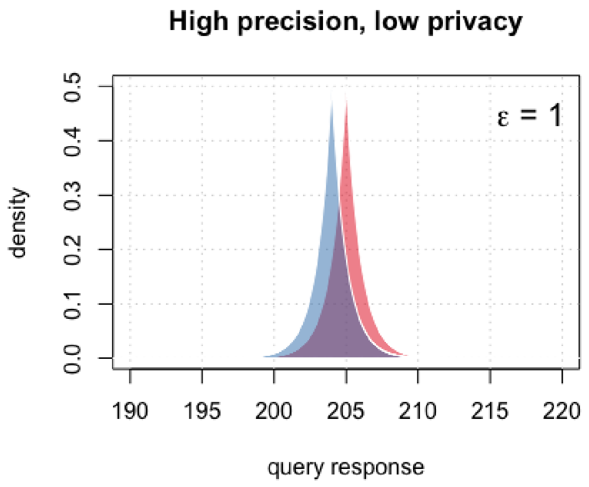

Fig. 2. Distribution of responses to the salary query with $\epsilon = 1$.

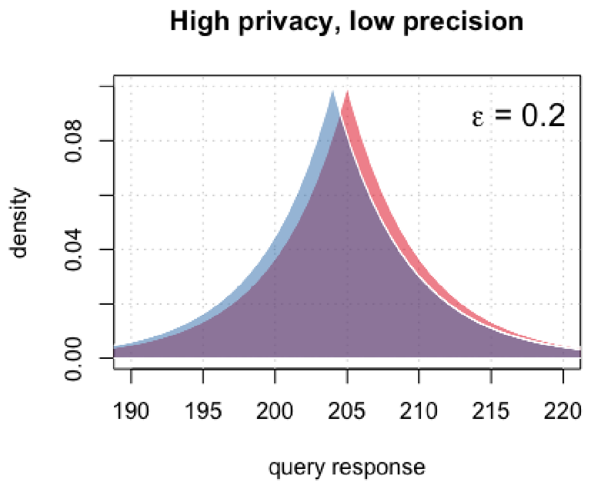

Fig. 3. Distribution of responses to the salary query with $\epsilon = 0.2$.

error of the randomized response approach from the previous subsection.

For this type of query, i.e asking for the number of rows which satisfy a property, differential privacy provides a suitable robust and mathematically rigorous approach which can still allow for precise inference about the database population as a whole. For other queries, such as arithmetic or rank queries, the required amount of noise will be a function of the query *sensitivity* which in turn relies of the underlying distribution of the variable with respect to the database. Unless we are willing to make distributional assumptions, a reasonable approach in many scenarios, then the required noise will be unbounded leading to useless privacy mechanisms with regards to reliable inference. These difficulties will be discussed further in Section 5. Although differential privacy with Laplacian noise is a natural and effective solution to the property-count query, we next consider an attack which allows for a large number of such queries. As we will shortly formalize, it is easy to see that differentially private systems will be vulnerable to such attacks based on the limit theorems of Statistics.

C. Protection Against Multiple Queries

If an adversary makes a query Q_i, hoping to discover the true response m, he will receive a response equal to $m + \delta_i$. Suppose the adversary makes N such queries, $Q_1, Q_2, \cdots Q_N$ and takes the average response $\bar{Q}$.

$$\bar{Q} = \frac{1}{N}\sum_{i=1}^{N} Q_i = m + \frac{1}{N}\sum_{i=1}^{N}\delta_i$$

This random variable, denoted $\bar{\delta} = \frac{1}{N}\sum_{i=1}^{N}\delta_i$ is subject to the Central Limit Theorem and the Law of Large Numbers, meaning that as N grows large $\bar{\delta} \sim N(0, \frac{2}{\epsilon^2 N})$ [10]. As the number of queries grows, the quantity $\bar{Q}$ converges in probability to the true response m. For instance, if $\epsilon = 1$ as it does in Figure 2, only 20 queries are required to know the value of m with 95% confidence. If $\epsilon = 0.2$ as in Figure 3, the same level of confidence can be achieved with a modest 500 queries. A seemingly obvious solution to this problem is to fix the pseudo-random procedure so that a query provides the same "random" response $m + \delta_i$ each time. To see why this approach fails, consider a series of *antithetic* queries given as follows.

- How many rows in the database satisfy property Q?
- How many rows in the database do not satisfy property Q?
- How many rows in the database satisfy property Q and property R?
- How many rows in the database satisfy property $\tilde{Q}$?

The first two queries look different, but in fact uncover exactly the same information. In the third query, if R is a property which is always satisfied for any row this also gives the same information. If R is almost always satisfied, then the query will give nearly the same result. If R is a complicated set of mathematical constraints, we find ourselves back at the beginning subject to a database reconstruction attack. The key realization is that *seeding* the pseudo-random responses is futile, since there are a number of related queries that can uncover the same information as the first, and identifying the complete set of related queries is a computationally infeasible practice. The final query in the list given above refers to a property $\tilde{Q}$ which is *correlated* with property Q. A response to this query will not fully uncover the answer to the first, but it provides some level of information nonetheless [11]. When two queries are completely unrelated, differential privacy can precede as in the previous subsection unharmed. Consider a database which stores information on height and weight of American women, and an adversary is interested in learning the true average weight w of this population. A direct query will produce the result $w+\delta_w$ where δ_w is a Laplacian random variable with variance e. A second query for the height of these women returns the response $h + \delta_h$ where δ_h again has variance e. At first, this does not appear to have given any additional information to the adversary about w. Suppose now that the adversary has access to an auxiliary study giving the relationship:

$$w = -45.7 + 0.63 \times h + \delta_r$$

where δ_r is random noise with a variance of 20. Using this information, the adversaries best guess for w is now

$$\hat{w} = \frac{w + \delta_w - 45.7 + 0.63(h + \delta_h) + \delta_r}{2}$$

which is an unbiased estimate for w with variance $0.7e + 10$. If the privacy guarantees are such that $\epsilon < 0.25$ then this estimate is an improvement. If ϵ is public information, then a weighted average approach can always lead to improved estimation. This example is important because it shows that any two queries, whether or not they initially seem related, can provide information about the other. Thus query systems

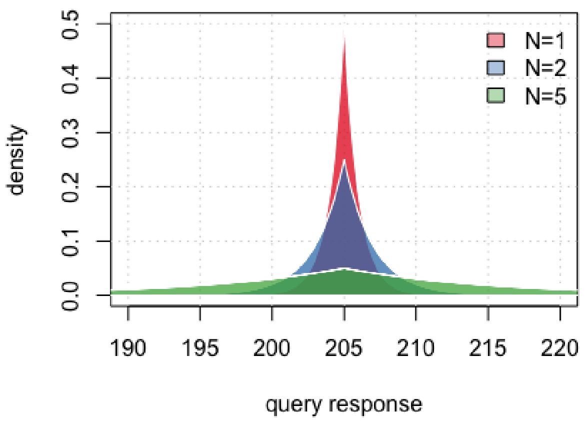

Fig. 4. Distribution of responses to the salary query with $\epsilon = 1$ as the number of query repetitions (N) increases.

which allow multiple queries should handle the noise injection carefully in order to maintain the privacy requirements.

A simple solution to the multiple query problem is to inject more noise for more multiple queries. If the queries are required to be listed in advance, and there are N such queries, this can be accomplished by generating response $m + \delta_i$ where δ_i is Laplacian error with parameter ϵ/N leading to a standard deviation of $\sqrt{2}N/\epsilon$. Figure 4 shows the distribution of responses, when the true response is $m = 205$, as the number of $\epsilon = 1$ differentially private queries increases $N = 1, 2, 5$. For small values of N, the result looks reasonable and precise inferences can still be made, but as N increases to just 5, the inference is bound to be necessarily imprecise. As privacy increases and ϵ gets smaller, it can become increasingly difficult to guarantee precise procedures for inference.

D. Strengths of Differential Privacy

Protecting the privacy of an individuals data while maintaining the statistical utility of a database has proven itself to be a difficult task. Although differential privacy has many challenges, it is nonetheless a powerful, useful and promising tool for many problems. The level of noise injection required for a query response is a direct product of the query sensitivity which we formally define now for the first time [7].

Definition: Consider a query function $g : D \to \mathbb{R}^d$. The L_1 sensitivity of the function f is

$$\begin{aligned}\Delta g &= \max_{(D,D')\in\Omega_1} \|g(D) - g(D')\|_1 \\ &= \max_{(D,D')\in\Omega_1} \sum_{i=1}^{d} |g(D)_i - g(D')_i|\end{aligned}$$

Where Ω_1 is the set of all database pairs (D, D') which differ by exactly one row.

A property-count query g_{pc} can be treated as a function mapping a database into the set of non-negative integers. For this type of query, the sensitivity reduces to $\Delta g_{pc} = \max |g_{pc}(A) - g_{pc}(B)|$ where B is the complete dataset and A is identical except for one missing row. This result can either be 0 (if the missing row lacks the property of interest) or 1 (if the missing row has the property of interest). The maximum over all possible omitted individuals clearly indicates that $\Delta g_{pc} = 1$. If a series of N such queries are made, this can be treated as a function g which maps each dataset D to the cartesian product of N sets of non-negative integers:

$$\{(z_1, \cdots z_n) \mid z_i \in \mathbb{Z}_+ \cup \{0\}\}.$$

If the omitted individual satisfies the property of each of the N queries, the L_1 sensitivity of this function is obviously $\Delta g = N$. It can be shown that for a general query function of sensitivity Δg, a noise injection level of $\epsilon/\Delta g$ is sufficient to guarantee ϵ-differential privacy. Indeed as we just illustrated, this general result captures the multiple query scenario discussed in the previous subsection.

A *histogram query* refers to the process of ”binning” or ”partitioning” responses into a number of categories [5]. Take for instance the following examples using a database of employees at a fictional company.

- Return a table giving the number of employees in each department.
- Return a table giving the number of employees in each tax bracket (based on salary).

At first glance, this appears to be a separate property-count query for each of the desired categories. If there are 20 departments, we might expect the sensitivity of such a query to be 20, a number which would place stress on the privacy/precision tradeoff. In fact, as long as the query forms a partition, i.e. the categories are *disjoint*, then adding (or removing) an individual will change exactly one of the counts and the sensitivity of such a query is $\delta g = 1$. Thus a large amount of information can be obtained, using a relatively small amount of noise injection to the response.

Other queries, as we will show in Section 5, can be more challenging. For now, we are content with the fact that differential privacy can be achieved in a robust and statistically powerful manner for a wide range of important queries. These ideas are advanced in the next section, where we discuss more complicated queries and how these can be implemented in a differentially private way.

IV. Differential Private Programming

The goal of *private programming* is to provide primitives which can be used in aggregate to perform complex functions while remaining confident that the result remained private, and to avoid the error-prone task of painstakingly proving the algorithms themselves private.

One implementation of this type of framework was centered around the Sub-Linear Query database (SuLQ) and was developed shortly *before* the formalization of differential privacy [12]. The SuLQ framework assumes that the total number of queries is smaller than the size of the database, a reasonable constraint for many large databases. From this a set of SuLQ primitives were derived that returned simple statistical queries with an added amount of noise, which was a random number distributed according to a mean zero normal distribution. Together these primitives constructed the SuLQ framework, which upon different invocations could be used to perform the following computations: k-means clustering, ID3 decision tree classification, principal component analysis, and many others. Since the formalization of differential privacy, it has been shown that many of these private programming algorithms can be modified to achieve differential privacy. We illustrate this idea in the following subsection using the example of the differentially private *k-means clustering* algorithm.

A. k-Means Clustering

The k-means clustering algorithm attempts to find a solution to the following problem: Given a collection of points $\{p_i\} \subset [0, 1]^d$, find clusters of points such that each cluster contains points that are mutually proximate. The algorithm in its standard, non-private implementation selects initial candidate means $\mu_1, ..., \mu_k$ chosen at random from the d-dimensional cube and performs the following update rule, which is iterated over for a fixed number of times, or until a convergence criterion is met. A solution is found when for a given cluster of points, the points minimizing the radii of each cluster are equal to the mean of the cluster.

- Partition the samples $\{p_i\}$ into k sets $S_1, ..., S_k$, associating each p_i with the nearest μ_j.
- For $1 \leq j \leq k$, set $\mu'_j = \sum_{i \in S_j} p_i / |S_j|$, the mean of the samples associated with μ_j.

The first step of the update rule works specifically with each individuals data to compute the nearest mean, rendering it useless in terms of privacy. However, the algorithm can be adapted in the following manner, keeping in mind that this variation was initially implemented using the SuLQ framework, and later was adapted to adding Laplace noise in order to provide differential privacy. To calculate the update sum it can be noticed that the sensitivity of a single point p_i affects at most one sum, and the sum can change by at most 1 in each of the d dimensions. The candidate means $\{\mu_1, ...\mu_k\}$ partition the space of $[0, 1]^d$, which can be likened to a standard histogram query and result in queries about the cardinality of the sets $\{|S_1|, ..., |S_k|\}$ maintaining differential privacy of individual points. Therefore, the primitives that perform the update rule will remain differentially private so long as the query sequence operates with total sensitivity of at most $d + 1$. If the update rule is set to run a fixed number of iterations N, ϵ-differential privacy can be obtained using query response $m + \delta, \delta \sim Laplace(\epsilon/((d+1)N))$. If the number of iterations is unknown, differential privacy can be achieved by increasing the amount of added noise with each computation using a *privacy budget*, a method discussed further in the next section [7].

Recent work has focused on the creation of a strongly-typed framework to provide differentially private primitives that can be used in conjunction to perform complex functions. One framework, called PrivInfer, uses a type-based framework that utilizes symbolic probability distribution to support provably differentially private Bayesian machine learning algorithms [13].

V. Practical Difficulties Concerning Differential Privacy

There are two primary weaknesses of using differentially private systems in practice. Differential privacy refers to the mathematical formalization of the promise that ”no additional harm will come to an individual based on his inclusion (or exclusion) in the database”. It is important to note that these concerns, substantial though they may be, are not directly related to differential privacy but deal more with the implementation of differentially private databases. Primarily, we have focused on achieving differential privacy via output noise injection, where the noise level depends on the query sensitivity. When the query-sensitivity is large, achieving differential privacy may be challenging or impossible in a way that maintains the desired statistical properties needed for precise inference. A second concern is that of the privacy budget implementation. There are already difficulties with maintaining an inferentially useful system which allows multiple queries *in theory*. In practice, these difficulties are amplified many times over.

A. Highly Sensitive Queries

Some useful queries, such as histogram queries, have a sensitivity of 1. For other queries, the sensitivity may be much larger. Consider the query which asks for the maximum salary of all individuals in the database and suppose that the current true response to this query is m. We consider a new individual with salary $10m$ and create a new database C with this addition. Clearly the sensitivity of this max-query is *at least* $10m - m = 9m$ which is already highly sensitive when m is large. Unless we know the salary of the richest man in the world, the sensitivity of such a query is unbounded. Many non-counting queries which request a numerical summary of a variable, such as mean or variance will be plagued by this same problem. If the range of a variable is unbounded, the sensitivity of a mean-query will also be unbounded [15]. In some instances, it may be reasonable to provide a truncation threshold, or upper bound to the response returned by a query. Adding boundaries to the universe of responses can guarantee that the query-sensitivity exists, but if the upper bound is much larger than a realistic response m the sensitivity can be enormous as it is in the salary example above, rendering the sensitivity result useless in practice. If the bound is tightened, inferences will be biased unless a censoring indicator is also returned by the query (i.e the response would be $> m$). This is a reasonable solution, as the statistical field of *survival analysis* is well equipped with the tools to analyze this kind of *censored data*. Unfortunately, the information given by a censoring mechanism may be difficult to analyze directly making privacy guarantees hard to analyze. Much like we saw with the query auditing suggestion, censoring itself can be disclosive of private information [14].

A more promising strategy, for a wide variety of cases, is to make distributional assumptions on the data and analyze these queries from a statistical perspective. For illustration, we consider an example using a hypothetical dataset containing information for the 500 employees of a small company. We begin by making a simple histogram query with sensitivity $\Delta g = 1$. Figure 5 shows the true response to a histogram query over the salary partition $[0, 25)$, $[25, 50)$, $\cdots$, $[250, \infty)$ in thousands of dollars. Figure 6 shows the noisy query returned to the user using the exponential privacy mechanism with $\epsilon = 1$.

Salary data is often fitted with a log-normal assumption, and the red dashed line in Figure 6 represents a log-normal distribution, with mean 74.9 and standard deviation 38.3, fitted to the noisy salary data using standard *censored likelihood* approaches. An estimate response to the query for the mean salary of this dataset would be the fitted value $\$74,900$. The maximum value, given that the sample size is 500, can be estimated as $\$267,000$ (the true max salary of this dataset was actually $\$271,450$). This procedure satisfies ϵ-differential privacy as a direct corollary of the histogram query since it can be done by an adversary outside the query system. While privacy guarantees can be made using this type of approach, the validity of the statistical inference is highly sensitive to the choice of distribution. For instance, if a Normal distribution was assumed for the salaries (a poor assumption given the skewness), the response to the maximum-salary-query is severely underestimated at $\$197,157$. Of course, not all datasets and variables can be appropriately analyzed using parametric distributions, especially if there are outliers in the dataset and inappropriate modeling choices can lead to incorrect or badly biased query responses.

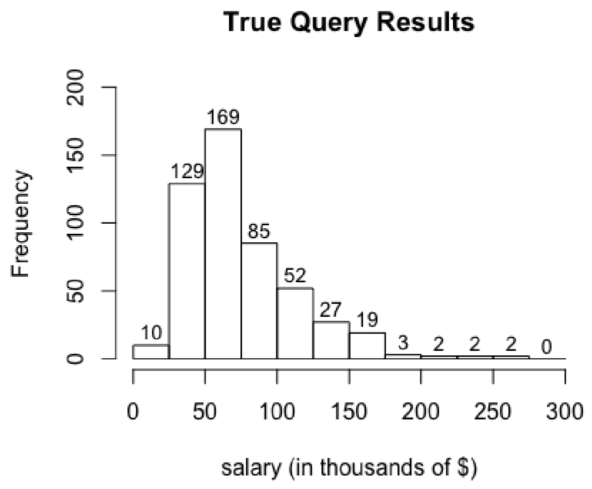

Fig. 5. True response to the histogram query for the hypothetical salary example.

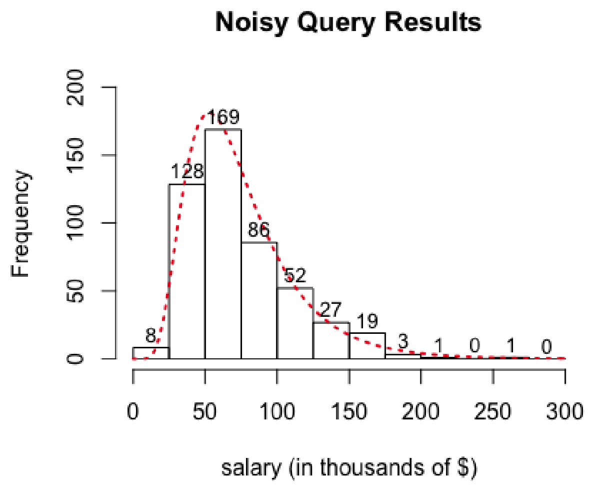

Fig. 6. Response to the histogram query with noise injection ($\epsilon = 1$) for the hypothetical salary example. Red curve represents a Log-normal distribution, fitted to the noisy query results using censored likelihood.

B. The Privacy Budget

Differential privacy is achieved primarily through output noise injection which is a function of query sensitivity. We showed in an earlier section that the N-query problem consisting of sequential queries $g_1, \cdots g_n$ can be treated as a single query g with L_1 sensitivity $\Delta g = \sum_{i=1}^{n} \Delta g_i$. Repeating the same property-count query N times, for instance, results in a sensitivity of N which requires a large amount of noise injection to protect each individuals privacy. This discussion was focused on the "theoretical how" of privacy protection against multiple queries. In this section we discuss the "practical how" of multiple query protection and the implied limitations. The first step towards accomplishing this is to introduce the idea of a *privacy budget* [7].

Suppose that the user (or database system) specifies in advance the total (or maximum) number of queries desired (or allowed). Denoting this quantity by N_{max}, each query response can be injected with Laplacian noise using the parameter $\epsilon = E/N_{max}$. Once N_{max} queries have been made, the user is cut off.

An alternative implementation places no restriction on the maximum number of queries, but injects more noise as the query session goes on. In particular, the first query Q_1 is answered at a $\epsilon = E/2$ differentially private level. Query Q_2 is answered with $\epsilon = E/4$ noise injection and in general Q_k is answer at the $\epsilon = E/2^k$ level. Thus each query uses half of the remaining privacy budget. While this doesn't technically induce a maximum number of queries made by the user, the query responses quickly become useless for practical inferential purposes. This idea can be generalized, by allowing each query to use some proportion $\theta \in (0, 1)$ of the remaining budget. This generalized level of noise injection can be shown to satisfy E-differentially private for any number of queries.

Procedure: Given a privacy budget E and query allowance $\theta \in (0, 1)$, answer the k^{th} query as

$$Q_k \leftarrow m + \delta_k$$

where δ_k is a Laplacian random variable with noise parameter $(1-\theta)\theta^{k-1}E$.

Since $\theta \in (0, 1)$, and the variance of the noise level increases by $\mathcal{O}(\epsilon^{-2})$, it is clear that each successive query will require more noise injection than its predecessor. This is required in order to maintain privacy requirements for a large number of queries. This is guaranteed to be a E-differentially private process for any number of queries N. The proof of this statement is a straightforward consequence of the *geometric series*.

Proof: This satisfies the privacy budget for any number of queries N because:

$$\begin{aligned}\sum_{k=1}^{N}(1-\theta)\theta^{k-1}E &= (1-\theta)E\sum_{k=1}^{N}\theta^{k-1}\\ &= E\frac{1-\theta}{\theta}\sum_{k=1}^{N}\theta^{k}\\ &\le E\frac{1-\theta}{\theta}\sum_{k=1}^{\infty}\theta^{k}\\ &= E\frac{1-\theta}{\theta}\frac{\theta}{1-\theta}\\ &= E\end{aligned}$$

Once the privacy budget E has been set, a user can decide what allowance θ to use, which defines the decay rate of the "usefulness" of each query. By choosing $\theta \approx 1$, the first query will be maximally useful, but successive queries will suffer from a large amount of noise injection. By setting $\theta \approx 0$, each query will receive a large amount of noise injection. Reasonable choices of the budget allowance parameter seem to be $\theta = 1/2$ or $\theta = 1/3$. Although there is, strictly speaking, no maximum number of queries allowed, there is an *effective* max query size which we can define as the number of queries before all but 1% of the budget has been exhausted. This effective query limit evaluates to 7 queries for $\theta = 1/2$ and 12 queries for $\theta = 1/3$.

This value E is known as the *personal* privacy budget since it applies separately to each user. In practical settings however, this may be insufficient. For instance, what is to stop the user from storing the query results and logging into the database system a second time (or third, fourth, etc.) and repeating the process. It is obvious that careful precaution must be taken on the back-end to avoid this, a challenging task for a data publisher who wishes the societal benefits of the dataset to be easily accessible. These concerns have led to the joke that once a user has exhausted his or her personal privacy budget we must "kill the user".

A much bigger problem emerges if we take this one step further and allow for collusion between a collection of adversaries [15]. If each of K adversaries are given a personal privacy budget E, then the collective privacy budget of the group is KE, an unacceptable breach of privacy when K is large. This leads naturally to the concept of a *Global* privacy budget G, which represents the privacy budget for the database system as a whole. Allocation of these global resources becomes a challenging task. If a first-come-first-serve approach is used, the entire budget can be rapidly depleted rendering the dataset useless. Another option is to partition the global budget G to a pre-determined set of K users so that the allocated personal budgets, E_k, sum to G. For particular cases, the implementation of a global privacy budget may be feasible, but it is clear that this solution seriously limits the statistical usefulness of a database in practice.

VI. Modern Advancements for Differential Privacy in Practice

A. *Differentially Private Query Languages*

There have been recent attempts to implement query languages which respond to queries in differentially private manner [16] [17]. The Privacy Integrated Queries (PINQ) system has been around since 2009 and provides differential privacy for counting queries using an augmented version of SQL. This language supports many standard database operations including a *join* operation which has some limitations for one-to-many and many-to-many join operations. In 2014, a refined version of the query language called weighted PINQ (wPINQ) addressed this limitation by supporting a general equijoin operation with a differential privacy guarantee. These earlier systems were based on the idea of global query sensitivity which leads to yet another implementation concern.

> *"a join has the ability to multiply input records, so that a single input record can influence an arbitrarily large number of output records."*

For this reason, it is ideal to work with *local* query sensitivity which places tighter bounds on the sensitivity Δg of a particular query instance. Determining the local sensitivity, especially for large databases, is a computationally infeasible problem possibly requiring billions of auxiliary queries for a single query of interest. In 2018, the notion of *elastic sensitivity* emerged as a computationally affordable upper bound for local sensitivity. The authors use this method of elastic sensitivity to build the query language FLEX, which they claim

- supports general equijoins and can be calculated efficiently using only the query itself and a set of precomputed database metrics,
- is compatible with *any* existing database,
- can enforce differential privacy for the majority of real-world SQL queries,
- and incurs negligible (0.03%) performance overhead [17].

In addition, the FLEX system takes a flexible approach to the treatment of the privacy budget, allowing for database managers to allocate resources as they see fit. The FLEX query system was used to answer 9862 actual queries (all queries to the real-world database during October 2016). The FLEX language was able to answer the majority of these queries with $0.0001\% - 10\%$ error while maintaining an impressive global 0.1-differentially private requirement. When no join operations were required to answer the query, nearly 75% of the queries were answered with less than 10% error. The query response accuracy improves even further when the query refers to a large population - roughly 99% of queries had less than 10% error when the effective population was at least $10,000$. Although differential privacy has a long way to go, these are important steps towards practical and publicly available differentially private frameworks.

B. *Noise Injection Strategies and the 2020 Census*

Differential privacy makes a promise to users that their existence in a database will cause them no additional harm.

Addition of noise at the query level is known as *output* noise injection and, done in the right way and magnitude, is sufficient to guarantee that a query system is differentially private. There are many strengths to this approach including its robustness to database reconstruction attacks. A naive approach based on straightforward application of a SAT solver is almost certain to fail. More intelligent approaches *may* be able to reconstruct a set of microdata which is mostly consistent with the original data, but the reconstruction will almost certainly differ from the original data.

A related approach involves the application of noise *before* the data is tabulated, a process known as *input* noise injection.There are two immediate downsides to this approach. The first, is that we lose flexibility in our privacy allowances. All users are given access to exactly the same noisy data, and all queries are subject to the same level of inaccuracy. With the geometric privacy-budget-allowance approach, the privacy budget resources can be spent sequentially so that more ”important” queries (to the user) are asked first. Another flaw is that input noise addition will not prevent an adversarial database reconstruction attack, although it does limit the reliability of such a reconstruction since the microdata will be noisy.

Another form of input noise injection, known as *swapping*, refers to the process of randomly exchanging row values within each column. Although this method has some nice features in practice, a large proportion of rows may need to be swapped in order to protect the privacy of each individual. This operation will have no effect on the query results when conducted at a high level. For instance, if we swap two rows in a database so that Alan Turing is now listed with brown eyes and John Von Neumann with blue eyes, the overall count of individuals in the database with blue eyes will remain unchanged. Alternatively, the query ”how many people in the database are from England and have blue eyes?” will now be incorrect by 1. Thus swapping also experiences a privacy/precision tradeoff. Swapping should be applied at lower levels of geography (such as City or ZipCode) to ensure precision but at high levels (such as Country or State) to ensure privacy.

The United States government used swapping as one of its primary methodologies for privacy protection in the 2010 census and plans to use it to some extent again in 2020 [1] [18]. In addition, the 2020 census is adopting a privacy-protection system which is heavily based on differential privacy. This methodology will protect census respondents from database reconstruction attacks, while ensuring that the resulting statistical inference is sufficiently accurate for its intended purpose.

VII. Concluding Remarks

Modern database owners face the unique and challenging problem of publishing and maintaining datasets which can be accurately mined for the betterment of society while simultaneously upholding an ethical obligation to protect the privacy of the individual. Differential privacy offers a *realistic* promise to protect the privacy of the individual while preserving the intended usefulness of the dataset. This framework offers a mathematically rigorous definition of privacy by requiring that the results of a statistical analysis be ”essentially indistinguishable” with the omission or addition of a single person. Although the differential privacy framework has shown promise, there are several important limitations both in theory and in practice. With the introduction of a formal framework for privacy protection, future research in the field of data-privacy now has a standard benchmark. The 2020 US census has already adopted a differential-privacy policy acknowledging its usefulness, despite the limitations. With continued and dedicated research to such an important field, the 2030 census may experience higher levels of data-privacy and statistical precision than ever before.

References

[1] Garfinkel, Simson L., John M. Abowd, and Christian Martindale. ”Understanding database reconstruction attacks on public data.” (2018).

[2] Marques-Silva, Joao, Inłs Lynce, and Sharad Malik. ”Conflict-driven clause learning SAT solvers.” Handbook of satisfiability 185 (2009): 131-153.

[3] Dwork, Cynthia, Frank McSherry, Kobbi Nissim, and Adam Smith. ”Calibrating noise to sensitivity in private data analysis.” In Theory of cryptography conference, pp. 265-284. Springer, Berlin, Heidelberg, 2006.

[4] M. Isaac and S. Frenkel, ”Facebook Security Breach Exposes Accounts of 50 Million Users”, The New York Times, 2008.

[5] Dwork, Cynthia, and Aaron Roth. ”The algorithmic foundations of differential privacy.” Foundations and Trends in Theoretical Computer Science 9, no. 3?4 (2014): 211-407.

[6] Dalenius, Tore. ”Towards a methodology for statistical disclosure control.” statistik Tidskrift 15, no. 429-444, 1977

[7] Dwork, Cynthia. ”A firm foundation for private data analysis.” Communications of the ACM 54, no. 1 (2011): 86-95.

[8] Van Erven, Tim, and Peter Harremos. ”Rnyi divergence and Kullback-Leibler divergence.” IEEE Transactions on Information Theory 60, no. 7 (2014): 3797-3820.

[9] Warner, Stanley L. ”Randomized response: A survey technique for eliminating evasive answer bias.” Journal of the American Statistical Association 60, no. 309 (1965): 63-69.

[10] Plya, G. ”On the central limit theorem of calculus of probability and the problem of moments.” Math. J., German 8, no. 3 (1920): e4.

[11] Zhu, Tianqing, Ping Xiong, Gang Li, and Wanlei Zhou. ”Correlated differential privacy: Hiding information in non-iid data set.” IEEE Transactions on Information Forensics and Security 10, no. 2 (2015): 229-242.

[12] Blum, Avrim, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. ”Practical privacy: the SuLQ framework.” In Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 128-138. ACM, 2005.

[13] Barthe, Gilles, Gian Pietro Farina, Marco Gaboardi, Emilio Jesus Gallego Arias, Andy Gordon, Justin Hsu, and Pierre-Yves Strub. ”Differentially private bayesian programming.” In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 16, pp. 6879. ACM, 2016.

[14] Garfinkel, Simson L. ”De-identification of personal information.” NISTIR 8053 (2015): 1-46.

[15] Haeberlen, Andreas, Benjamin C. Pierce, and Arjun Narayan. ”Differential Privacy Under Fire.” In USENIX Security Symposium. 2011.

[16] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pages 19?30. ACM, 2009.

[17] Johnson, Noah, Joseph P. Near, and Dawn Song. ”Towards practical differential privacy for SQL queries.” Proceedings of the VLDB Endowment 11, no. 5 (2018): 526-539.

[18] United States Census Bereau, ”Data Portection and Privacy Program”, http://www.census.gov, 2014.