The Analytic Framework for Data: A Cryptographic View

Cynthia Dwork, Microsoft Research

October 15, 2013

Abstract

Among, and enabling, the breathtaking advances of modern cryptography is a methodology for defining and proving security. Central to this methodology is the practice of circumscribing all potential attacks with an "adversary" whose powers – computational and informational – and goals (what does it mean to "break" the system?) are spelled out. Differential privacy, a definition of privacy tailored to statistical data analysis, emerged from this intellectual tradition.

1 Introduction

Propose. Break. Propose Again. So Pre-Modern cryptography cycled. An encryption scheme was proposed; a cryptanalyst broke it; a modification, or even a completely new scheme was proposed. Nothing ensured that the new scheme would in any sense be better than the old. Among the astonishing breakthroughs of modern cryptography is the methodology of rigorously defining the goal of a cryptographic primitive – what it means to break the primitive – and providing a clear delineation of the power - information or computational ability - of the adversary to be resisted. Then, for any proposed method, one proves that no adversary of the specified class can break the primitive. If the class of adversaries captures all feasible adversaries, the scheme can be considered to achieve the stated goal.

This does not mean the scheme is invulnerable, as the goal may have been too weak to capture the full demands placed on the primitive. For example, when the cryptosystem needs to be secure against a passive eavesdropper the requirements are weaker than when the cryptosystem needs to be secure against an active adversary that can determine whether or not a ciphertext is well-formed (such an attack was successfully launched [1] against PKCS#1). In this case the goal may be reformulated to be strictly more stringent than the original goal, and a new system proposed (and proved). This strengthening of the goal converts the propose-break-propose again cycle into a path of progress.

1.1 Toward Articulating a Privacy Goal

This is often best approached by articulating what it means to "break" a system. Let us look at some examples.

Linkage Attacks. The litany of these attacks includes re-identifying the medical records of Governor William Weld among anonymized medical encounter data by linkage with voter registration records [34] and identifying a Netflix user among anonymized training data for a competition on recommendation systems by linkage with the Internet Movie Database (IMDb) [28].

The Statistics Masquerade. Privacy problems do not disapper if we give up on a release of "anonymized" microdata and turn to the release of statistics. For example, the *differencing attack* exploits the relationships between certain pairs of statistics, such as:

1. The number of Microsoft Employees with the sickle cell trait; and
2. The number of Microsoft Employees, other than the director of the Silicon Valley Laboratory, with the sickle cell trait.

When taken together, these two statistics, each of which covers a very large set of individuals, reveal the sickle cell status of the director. Such dangerous pairs of queries are not always so easy to spot; indeed, if the query language is sufficiently rich the question of whether two queries pose such a threat is undecidable.

A more general adversarial strategy may be called the "Big Bang" attack. Given a large data set, the attacker focuses on a relatively small subset, say, members of his extended family, of some size k. For concreteness, let us set $k = 128$. The attacker's goal is to learn a single private bit – not necessarily the same bit – about each member of the extended family. For example, the attacker may wish to know if Aunt Wilma, who has two children, has had more than two pregnancies, and to know if Uncle William has a history of depression, and so on. Clearly, by asking k "counting queries," each describing exactly one member of the extended family and the property in question, eg, "How many people with the following identifying characteristics [description of Aunt Wilma and only Aunt Wilma] have had at least three pregnancies?" the attacker can learn the desired bits. But suppose the attacker does not receive perfectly accurate answers. Can introducing small inaccuracies to the query responses protect the family's privacy? Intuitively this approach seems perfect: it renders useless any query about an individual, while not significantly distorting "statistical" queries whose answers are expected to be fairly large.

The degree to which small distortions can protect against arbitrary counting query sequences depends on the size of "small" compared to the number of queries. For example, there is a sequence of 128 counting queries with that following property: If the errors introduced are always of magnitude at most 1, then the adversary can reconstruct at least 124 of the 128 private bits. If the errors have magnitude bounded by 3, the number accurately reconstructed is still at least 92. With these same bounds on E, taking $k = 256$, the adversary can correctly reconstruct at least 252 and 220 bits, respectively.

The general form of the bound is: If the mangitudes of the errors are all bounded by E, then at least $k - 4E^2$ bits can be correctly reconstructed [12][1]. As we will see, this is approximately the "right" answer, in that noise of magnitude $E > \sqrt{k}$, correctly generated, is protective (and satisfies differential privacy).

The Big Bang attack is concrete. It gives a simple and computationally very efficient method by which information released by a disclosure control method that yields accurate answers to a relatively small number of apparently statistical queries can be abused to compromise privacy. The basic result is also very robust; with slight changes in bounds other attacks with similar outcomes can be launched using *random* queries, even if more than one fifth of the responses are completely arbitrary [9]. In some cases the attack can be launched against the enitre database (*i.e.*, $k = n$).

The Kindness of Strangers. Now that the era of "Big Data" is upon us, personal information – our searching, traveling, purchasing, entertainment histories – flows from one individual to another

[1]The attack involves computation of a Fourier transform and does not require knowledge of E.

via statistical learning systems. The set of search hits that receive clicks from one user affects which hits are shown to the next user; our presence on the road affects congestion which in turn affects route recommendations; recommendation systems suggest products based on observed paired purchases; Netflix recommends movies based on preferences of "similar" viewers. Can these flows be used to compromise privacy?

Astonishingly, despite any potential adversary's tremendous uncertainty regarding the data set, such attacks are known. The currently known examples require a small amount of *auxiliary information*, that is, information known to an attacker from a source – in this case something as simple as the purchaser's public blog – other than the data set. For example, a blog about a recent purchase, taken together with the vendor's (*e.g.*, Amazon's) continually changing public lists of "similar items," can reveal purchases not disclosed in the blog.

Smoking Causes Cancer. Defining a *query* to be a function mapping data sets to some output range, we can view everything discussed so far – the production of microdata, statistics, predictors, classifiers, and so on – as queries. A user's interaction with a data set can be viewed as receiving responses to queries, and a natural attempt to articulate a privacy goal tries to relate what is known about a member of the data set before, versus after, obtaining the response to a query or sequence of queries. Ideally, nothing would be learned about an individual from such an interaction.

This turns out to be unachievable if the responses are *useful*, in that they teach us things we did not know [6, 13]. We would like to learn facts such as "smoking causes cancer," but in doing so, our views and beliefs about individuals whom we know to smoke will change; for example, we will revise our predictions about their health. On the other hand, statistical analysis is meaningless without this type of *generalizability* – the whole point of a statistical database is to learn useful facts like "smoking causes cancer," not just for the participants in the study but for human beings in general. Our definition of privacy must take into account this desired utility.

Framing Our Goal: In/Out vs Before/After The "Smoking Causes Cancer" example shows the technical difficulty of the "Before vs. After" approach to framing a privacy goal.

The key insight is that, if the database teaches that smoking causes cancer, the bad (higher insurance premiums) and good (joins smoking cessation program) consequences for an individual smoker will be incurred *independent of whether or not the particular smoker is in the database.* This suggests a new privacy goal: to ensure that, by participating in a data set, one will be no worse off than one would be had one declined to participate. This is the heart of differential privacy.

Informed by our examples of attacks, we want this "In vs. Out" privacy guarantee to hold regardless of the sources of auxiliary information – detailed information about family members and co-workers, blogs, other data sets, product recommendations, *etc.* – to which an attacker may have access. Differential privacy will be such a guarantee.

1.2 An Ideal Scenario

Most of the literature on differential privacy assumes an ideal scenario in which the data are all held by a trusted and trustworthy *curator*, who carries out computations on the entire data set and releases the results to the data analyst. Not to put too fine a point on it, the *data* remain secret, the *responses* are published.

In reality, the data may not all reside in the same place – for example, the analyst may wish to study the combined medical records of multiple hospitals (and the hospitals don't want to share their data with one another), or the data may reside, encrypted, in a semi-trusted cloud, where the cloud is trusted to keep data intact and to run programs, but it is desired that the cloud operator not have access to unencrypted data. For these situations, cryptography comes to the rescue. For example, the first may be addressed through *secure multiparty computation* [31], and the second through *functional encryption* [4]. The role of cryptography in these cases is to abstract away the details and ensure that the system *looks just like*, or emulates, the ideal scenario.

Privacy-preserving data analysis is difficult even in the ideal scenario, but of course any real implementation of differential privacy – whether in differentially private generation of synthetic data that are released to the public or in differentially private query/response systems – questions of physical security of the data do not go away and must be addressed with additional other technology.

1.3 Adversaries

Who are the "adversaries" and what motivates them? To what kind of information do they have access? Do they collude, intentionally or accidentally? Here it seems we are limited only by our imagination. We list a few examples.

An abusive and controlling partner has enormous auxiliary information about the victim, including dates and details of abuse. Privacy of medical and police records may be a question of life or death in such a situation.

Snake oil salesmen who prey on the desperate are financially motivated to find very sick individuals. Purchasing, through an online advertising system, the ability to track individuals based on the issuing of certain search queries could be very lucrative, and very easy.

Blackmailers are motivated to find the unfaithful, for example, by analysis of telelphony and mobility records.

Learning the reading preferences of an employee or a perspective employee, via recommendation systems, can enable discrimination, or can inhibit intellecutal exploration.

A thief, observing patterns of power consumption through poorly anonymized smartgrid data, learns good times to break into homes.

Medical insurance companies wish to charge higher rates for customers with less healthy, or more risky, eating, exercise, and sexual habits, which may revealed by purchase, search, and advertising click histories.

A member of a middle-class community might find her relationships with her neighbors significantly altered were they to learn that despite her modest living style she has a seven-figure income.

2 Differential Privacy

In English, differential privacy says that the *distribution* on the outcome of any analysis is essentially unchanged independent of whether any individual opts in to or opts out of the data set. "Essentially" is formalized by parameter, usually called epsilon (ε), measuring privacy loss. Here, the distribution is taken over the coin tosses of the algorithm that is protecting privacy.

Before formalizing the definition, we note some properties of the guarantee. First and foremost, it says that the property of being differentially private depends only on the algorithm for carrying out the analysis – something that the data curator, or "good guy," controls. Thus, if an algorithm is differentially private then it remains differentially private no matter what an adversarial data analyst knows – including to which other data sets he or she has access. So differentially private algorithms automatically protect against linkage attacks. Differential privacy even guarantees that, if the analyst knows that the data set is either D or $D' = D \cup \{p\}$, the outcome of the analysis will give at most an ε advantage in determining which of D, D' is the true data set. More precisely, if the adversary begins with a prior distribution in which the two data sets are equally likely, the posterior probabilities can change by at most an e^{ε} factor. (When $\varepsilon \ll 1, e^{\varepsilon} \approx (1+\varepsilon)$; $e^{1/10} \approx 1.1052$).

2.1 Formal Definition of Differential Privacy

A database is modeled as a collection of *rows*, with each row containing the data of a different individual. Differential privacy will ensure that the ability of an adversary to inflict harm (or good, for that matter) – of any sort, to any set of people – should be essentially the same, independent of whether any individual opts in to, or opts out of, the dataset. This is done indirectly, simultaneously addressing all possible forms of harm and good, by focusing on the probability of any given output of a privacy mechanism and how this probability can change with the addition or deletion of any one person. Thus, we concentrate on pairs of databases (D, D') differing only in one row, meaning one is a subset of the other and the larger database contains just one additional row. (Sometimes it is easier to think about pairs of databases of the same size, say, n, in which case they agree on $n-1$ rows but one person in D has been replaced, in D', by someone else.) We will allow our mechanisms to flip coins (in fact, it is required); such algorithms are said to be *randomized*.

Definition 1. [6, 8] *A randomized mechanism M gives* $(\varepsilon, 0)$-differential privacy *if for all data sets D and D' differing on at most one row, and all $S \subseteq Range(M)$,*

$$\Pr[M(D) \in S] \quad \leq \quad e^{\varepsilon} \times \Pr[M(D') \in S],$$

where the probability space in each case is over the coin flips of M.

In other words, consider any possible set S of outputs that the mechanism might produce. Then the probability that the mechanism produces an output in S is essentially the same – specifically, to within an e^{ε} factor – on any pair of adjacent databases. This means that, from the output produced by M, it is hard to tell whether the database is D which, say, contains my data, or the adjacent database D', which does not contain my data. The intuition for privacy is: if you can't even tell whether or not the database contains my data, then you can't learn anything about my data.

2.2 Properties of Differential Privacy

In addition to immunity against linkage attacks, differential privacy offers several other benefits. We briefly describe a few of these.

Addresses Arbitrary Risks. Any data access mechanism satisfying differential privacy should satisfy all concerns one might have about the leakage of her personal information, regardless of any auxiliary information – other databases, newspapers, websites, and so on – known to an adversary:

even if the participant removed her data from the data set, no outputs (and thus consequences of outputs) would become significantly more or less likely. For example, if the database were to be consulted by an insurance provider before deciding whether or not to insure a given individual, then the presence or absence of *any* individual's data in the database will not significantly affect her chance of receiving coverage. Protection against arbitrary risks is, of course, a much stronger promise than the often-stated goal in sanitization of protection against re-identification. And so it should be! Without re-identifying anything, an adversary could still learn that a neighbor, observed to have been taken to the emergency room (the ambulance was seen), has one of only, say, three possible complaints[2].

Quantification of Privacy Loss. Differential privacy is not binary; rather, privacy loss is *quantified* by the maximum, over all $C \subseteq \text{Range}(\mathcal{M})$, and all adjacent databases D, D', of the ratio

$$\ln\left[\frac{\Pr[M(x) \in C]}{\Pr[M(x') \in C]}\right]. \tag{1}$$

In particular, $(\varepsilon, 0)$-differential privacy ensures that this *privacy loss* is bounded by ε. This quantification permits comparison of algorithms: given two algorithms with the same degree of accuracy (quality of responses), which one incurs smaller privacy loss? Or, given two algorithms with the same bound on privacy loss, which permits the more accurate responses?

Automatic and Oblivious Composition. Given two differentially private computations, on the same or on different, possibly overlapping, databases, where one is $(\varepsilon_1, 0)$-differentially private and the other is $(\varepsilon_2, 0)$-differentially private, the cumulative privacy loss incurred by participating in (or opting out of) (both) database(s) is at worst $\varepsilon_1 + \varepsilon_2$. This is true even if the responses are generated obliviously of one another. This also teaches us one way to cope with high demand; for example, to ensure a cumulative loss bounded by ε^* over k computations, it is enough to ensure that each computation is $(\varepsilon^*/k, 0)$-differentially private. Composition bounds are what allow us to reason about cumulative privacy loss of complex differentially algorithms built from simple differentially private primitives (see [2] *et sequelae*). This "programmability" enables the construction of differentially private programming platforms [25, 33].

Group Privacy. Every $(\varepsilon, 0)$-differentially private algorithm is *automatically* $(k\varepsilon, 0)$-differentially private for groups of size k, for all k. This automatically protects small groups, such as families. It will not necessarily offer protection for large groups, and indeed it should not! If two databases differ significantly, their statistics are expected to change, and this should be observable if the databases are to be useful.

2.3 Achieving Differential Privacy

The differential privacy literature contains many astonishingly beautiful and powerful algorithmic techniques, some of which have given impressive results even on data sets as small as 70 records [18]. For the most part, we will confine ourselves in this chapter to some simple techniques that, nonetheless, have non-trivial applications; the power of these techniques is illustrated in Section 3.

Differentially private algorithms hide the presence or absence of a single row. Consider a real-valued function f. The (worst-case, or global) *sensitivity* of f is the maximum absolute value by

[2]two of which, say, a broken limb and heart attack, might be ruled out when the neighbor is seen the next day, leaving only "panic attack."

which the addition or deletion of a single database element can change the value of f:

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

where the maximum is taken over all pairs of *adjacent* databases. For vector-valued functions we extend this to the L_1-norm:

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

Speaking intuitively, Δf is the worst case difference that a differentially private algorithm for the function f will have to "hide" in order to protect the presence or absence of an individual.

Definition 2 (The Laplace Distribution). *The Laplace Distribution (centered at 0) with scale b is the distribution with probability density function:*

$$Lap(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right).$$

The variance of this distribution is $\sigma^2 = 2b^2$. We will sometimes write $Lap(b)$ to denote the Laplace distribution with scale b, and will sometimes abuse notation and write $Lap(b)$ simply to denote a random variable $X \sim Lap(b)$.

The Laplace distribution is a symmetric version of the exponential distribution.

We will now define the *Laplace Mechanism.* As its name suggests, the Laplace mechanism will simply compute f, and perturb each coordinate with noise drawn from the Laplace distribution. The scale of the noise will be calibrated to the sensitivity of f (divided by ε).

Definition 3 (The Laplace Mechanism). *Given any function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, the Laplace mechanism is defined as:*

$$M(D, f(\cdot), \varepsilon) = f(D) + (Y_1, \ldots, Y_k)$$

where Y_i are i.i.d. random variables drawn from $Lap(\Delta f/\varepsilon)$.

Theorem 4. *The Laplace mechanism preserves $(\varepsilon, 0)$-differential privacy.*

Example 5. *Counting Queries Queries of the form "How many people in the database are over six feet tall?" have sensitivity $\Delta f = 1$, since the presence or absence of any individual in D can affect the true answer by at most 1. Thus, the Lapalce mechanism will return the true count perturbed by a random draw from $Lap(1/\varepsilon)$.*

One way to handle $k > 1$ counting queries is via composition: by running each individual query with parameter ε/k we ensure that the cumulative privacy loss due to k queries is bounded by $k \cdot \varepsilon/k = \varepsilon$.

A second approach permits us to take advantage of the special properties of the particular set of counts we wish to compute, which may have lower senstivity than the worst case $\Delta f = k$, leading to better accuracy for the same privacy loss. An extreme case is illustrated in Example 6 described next.

Example 6. *Histograms In a histogram query, the universe of possible database rows is partitioned into a fixed set of bins, say k, so that every database row belongs it exactly one bin. The true answer to the histogram query H when the database is D is, for each of the k bins in H,* *the number of rows in D that are in the given bin. For example, the bin may be income ranges $[0, 25K), [25K, 50K), \ldots, [\geq 1,000,000]$ for the year 2011, so the query is asking about the distribution on incomes for the sample of the population that comprises database D. The sensitivity is a histogram query is 1, since the addition or deletion of one individual can change the count of at most one bin, and that change will have magnitude at most 1. Thus $||H(D) - H(D')||_1 \leq 1$ for all adjacent D, D'. Theorem 4 says that $(\varepsilon, 0)$-differential privacy can be achieved by adding independently generated draws from $Lap(1/\varepsilon)$ to each output of $H(D)$. Compare this to the accuracy we would have obtained naively, by viewing the histogram query as k independent counting queries (one per bin) and applying the composition result mentioned in Section 2.2, which would have suggested adding noise drawn from $Lap(k/\varepsilon)$ to the count for each bin – a factor of k worse than what we get by thinking carefully about sensitivity!*

The Laplace Mechanism provides *one* method for ensuring $(\varepsilon, 0)$-differential privacy for any value of $\varepsilon > 0$. It does not necessarily give the *best* method for every setting. For example, the method gives poor responses if we seek answers to a superlinear (in the size of the database) number of queries, but other algorithms [3, 32, 20, 15, 18] give meaningful responses even for a number of queries that grows *exponentially* in the size of the database! In Section 3 we briefly mention experimental results for computation of marginals using one of these algorithms.

Differentially private algorithms will typcially be composed of several steps, and the Laplace mechanism is frequently employed at one or more of these individual steps. It is therefore an important *primitive*, or building block. We will see an example of extensive use of this primitive in constructing differentially private probability distributions in Section 3.

The Exponential Mechanism. Differential privacy can be ensured for discrete output ranges by the *Exponential Mechanism* [26]. This mechanism uses a computation-specific *quality function* mapping (data set, output) pairs to a real number. The exponential mechansim assigns probabilities to outputs that grow exponentially in the quality, and then selects an output according to the resulting probability distribution. To ensure that the resulting distribution is not too sensitive to the presence or absence of any individual, before computing the output distribution the quality is divided by the sensitivity of the quality function – that is, the maximum over all outputs o and all pairs of adjacent databases D, D', of $|q(o, D) - q(o, D')|$. The exponential mechanism is another very important primitive. We will see an example of its use in Section 3 in selecting a "best" (or perhaps just sufficiently good) distribution from a family of distributions.

The Gaussian Mechanism. What about Gaussian noise? This distribution may be preferable to the Laplace distribution, due to its higher concentration. The addition of Gaussian noise yields a relaxation, called (ε, δ)-differential privacy, described in the next section. Roughly speaking, (ε, δ)-differential privacy ensures that with probabiltiy $1 - \delta$ the privacy loss is bounded by ε. By taking δ to be "cryptographically small[3]" we get a very strong and robust guarantee enjoying all the properties described in Section 2.2.

Redefining sensitivity to be the maximum L_2 difference $||f(D) - f(D')||_2$ on adjacent database D, D' (rather than the L_1 difference $||f(D) - f(D')||_1$ we have discussed until this point), we obtain the following theorem for the *Gaussian Mechanism*:

[3]That is, δ grows more slowly than the inverse of any polynomial in the size of the database.

Theorem 7. *Let f be a function mapping databases to $\mathbb{R}^k$, and let Δ denote the L_2-sensitivity of f. The "Gaussian Mechanism" that adds i.i.d. noise drawn from $\mathcal{N}(0, (\ln 1/\delta)/\varepsilon^2)$ to each of the k coefficients of f is (ε, δ)-differentially private.*

Like the Laplace Mechanism, the Gaussian Mechanism is an important primitive, especially in geometric algorithms for ensuring differential privacy [19, 29].

2.4 Relaxations

The literature also contains several relaxations of differential privacy. Roughly speaking, (ε, δ)-*differential privacy* permits the condition in Definition 1 to fail with probability δ. Definition 1 is just the special case of Definition 8 in which $\delta = 0$, explaining the presence of the second parameter.

Definition 8. [5, 11, 2, 7] *A randomized mechanism M gives (ε, δ)-*differential privacy *if for all data sets D and D' differing on at most one row, and all $S \subseteq Range(M)$,*

$$\Pr[M(D) \in S] \quad \leq \quad \exp(\varepsilon) \times \Pr[M(D') \in S] + \delta,$$

where the probability space in each case is over the coin flips of M.

Example 9. *Understanding the subtleties of a definition is important, so let us consider what would happen if we allowed δ to be large, say, $1/100$? This would be a foolish choice: for a database of size n it would permit an algorithm that releases, with no modification whatsoever, a randomly selected $n/100$, or 1%, of all database rows!*

Beyond supporting the addition of Gaussian noise, this relaxation is also useful in differentially private programming. For example, suppose we have two methods for differentially private release of a given statistic, say, the median income. The first method, A, always maintains $(\varepsilon, 0)$-differential privacy, but has poor accuracy on some inputs; the second method, B, has excellent accuracy, but its privacy loss exceeds ε on pathological inputs of a certain type, and only on these pathological inputs. We can use a differentially private test to determine whether it is safe to use Algorithm B on the given data set; but even if designed correctly there will be some very small probability, say, γ, that the test will erroneously indicate it is safe to use method B, potentially yielding probability γ of a large privacy loss. The best we can do in this case is to acheive (ε, γ)-differential privacy. We can make γ suitably small by designing the test with an extremely small probability of error.

A more advanced composition analysis than that mentioned in Section 2.2 shows that the composition of k mechanisms, each of which is (ε, δ')-differentially private, is, for all $\delta > 0$, at most $(\sqrt{2k \ln(1/\delta)}\varepsilon + k\varepsilon(e^\varepsilon - 1), k\delta' + \delta)$-differentially private [15]. When $\varepsilon \leq 1/\sqrt{k}$, this replacing of $k\varepsilon$ by roughly $\sqrt{k}\varepsilon$ can represent a very large savings, translating into much improved accuracy.

The proof of this advanced composition result exploits the fact that *privacy loss is a random variable whose expectation is small.* In particular, the expected loss of an $(\varepsilon, 0)$-differentially private algorithm, for $\varepsilon < 1$, is less than $2\varepsilon^2$. Further exploration of this observation leads to an incomparable relaxation, Concentrated Differential Privacy [14]. Roughly speaking, concentrated differential privacy ensures that privacy loss is tightly concentrated about its expectation. For the case of k counting queries, the addition of noise drawn from $\mathcal{N}(0, 1/k\varepsilon^2)$ ensures that the expected cumulative privacy loss is $\varepsilon^2/2$ and, for all m, the probability of privacy loss $m\varepsilon$ is bounded by $e^{-m^2/2}$.

This, then, is the sense in which the Big Bang attack gave us the right answer: ensuring some form of ε-differential privacy for k counting queries can be achieved with an expected $\sqrt{k}$ noise per query; matching, up to a constant factor, the lower bounds learned from the Big Bang attack. In fact, by careful coordination of noise values we can answer many, many more queries at little additional cost in accuracy [3, 32, 20, 18].

Both (ε, δ)-differential privacy and Concentrated Differential Privacy enjoy all the properties of differential privacy listed in Section 2.2.

2.5 An Aside.

Randomized Response [37], a well-known technique from the social sciences used to survey respondents about embarassing or illegal behavior, also provides differential privacy[4]. It is instructive to compare Randomized Response to the Laplace mechanism. For a single query "How many people in the data set ingested a controlled substance in the past week?" Randomized Response will yield an error on the order of $\sqrt{n}$, while the Laplace mechanism will yield an error on the order of $1/\varepsilon$, which is a constant independent of n.

What about multiple queries? Suppose we have a database with a single "sensitive" binary attribute, and that attribute is recorded using Randomized Response. In this case the population can be sliced and diced at will, and privacy of this single attribute will be maintained. In contrast, the Laplace and Gaussian mechanisms appear to cease to give meaningful responses after just under n^2 queries[5]. In this special case, randomized response is preferable once we require answers to a linear number of queries. Unfortunately, Randomized Response generalizes poorly to multiple attributes; it does *not* give a solution to the problem of answering many arbitrary counting queries on multi-attribute data.

3 Two Experimental Results

Since its inception, differential privacy has been the subject of intensive algorithmic research. OnTheMap, a differentially private US Census bureau web-based mapping and reporting application that shows where people work and where workers live, and provides companion reports on age, earnings, industry distributions, and local workforce indicators, satisfies *probabilistic differential privacy* [24]. While interactive, responding to queries issued by users of the site, the system gives exact answers computed from a differentially privately generated *synthetic data set* constructed from US census data. We can think of a synthetic data set as a collection of records with the same structure as real records, so that, for example, off-the-shelf software running on the original data set could also run on the synthetic data set. Given a (public) set $\mathcal{Q}$ of queries and a (private) database D, the goal is to produce a synthetic data set y with the property that for all $q \in \mathcal{Q}$, $q(y)$ yields a good approximation to $q(D)$.

As one might (by now) expect, it is impossible to simultaneously preserve any reasonable notion of privacy and to release a synthetic data set that answers "too many" queries with "too much"

[4]One version of randomized response goes as follows. Fix a specific yes/no question. The subject is told to flip a coin. If the outcome is heads, the subject answers the (yes/no) question honestly. If the outcome is tails, the subject flips a second coin and answers yes or no depending on the outcome of the second coin. This version of Randomized Response is $(\varepsilon, 0)$-differentially private for $\varepsilon = \ln 3$.

[5]In fact, correlations between noisy responses can be exploited to extract surprisingly accurate approximate answers *on average*, even for a superpolynomial number of queries [29].

accuracy. There are also considerations of *computational complexity*, that is, the computational difficulty of creating a synthetic data set with the desired properties. Two factors come into play here: the size of the set $\mathcal{Q}$ of queries for which the curator promises correct answers, and the size of $\mathcal{U}$, the *universe* of possible data items. For example, if we wish to describe humans by their DNA sequences, the size of the universe is exponential in the length of the DNA sequence; if instead we describe the humans in our data sets by 6 binary attributes, the size of the universe is only $2^6 = 64$. Although theoretical results suggest formidable computational barriers to building synthetic data sets for certain large $\mathcal{Q}$ or a large $\mathcal{U}$ cases [10, 36], the literature also contains some counterpoints with very encouraging experimental validation. We give two examples.

3.1 The MWEM Algorithm

The *MWEM* algorithm [18] optimizes an offline variant [17] of the Private Multiplicative Weights update technique [20]. A description of the techniques involved in these works is, unfortunately, beyond the scope of this book. Given a (public) set $\mathcal{Q}$ of queries and a (private) database D, the algorithm produces a synthetic data set y with the property that for all $q \in \mathcal{Q}$, $q(y)$ yields a good approximation to $q(D)$.

Tables of Marginals are tables that answer counting queries of a special form. The universe $\mathcal{U}$ of possible database elements are d-bit strings, representing, for each individual, the values of d binary attributes. A k-way marginal is specified by a set S of k of these d attributes. There are $\binom{d}{k}$ k-way marginals. MWEM was evaluated on the sets of all 3-way marginals for three datasets discussed in [16], for several values of $\varepsilon \in [0, 1]$. That is, $\mathcal{Q}$ is the set of all $\binom{d}{3}$ 3-way marginals. The smallest dataset consisted of only 70(!) six-attribute records. Of the $2^6 = 64$ possible settings of these bits, 22 appeared in the data set (so the continency table had 22 non-zero entries).

In each case, the synthetic dataset was evaluated by computing the relative entropy, or Kullback-Leibler (KL) divergence with the real data set. The resulting measurements were compared with reports in the literature [16] that, roughly speaking, capture the best that can be done non-privately[6].

Remarkably, even on the smallest data set the relative entropy closely approaches the ideal when ε reaches about 0.7. For two of the other datasets (665 records, 8 attributes, 91 non-zero cells; 1841 records, 6 attributes, 63 non-zero cells), the differentially private algorithms *beat* the non-private bounds once $\varepsilon \approx 0.7$ and $\varepsilon \approx 0.5$ respectively.

3.2 DP-WHERE

In this section we describe DP-WHERE [27], a differentially private version of the WHERE (Work and Home Extracted REgions) approach to modelling human mobility based on cellphone Call Detail Records [23]. For each individual, simultaneously, DP-WHERE protects *all* Call Detail Records in the data set; this is known in the literature as *user* level privacy (here "user" refers to a telephone user, not the data analyst)[7].

[6]Any set of marginals determines the maximum likelihood estimator (MLE) $\hat{p}$, which is the unique probability distribution in the model encoded by the given set of marginals that makes the observed data set D the "most likely" sample to have been observed. The bounds on KL divergence for the non-private case come from the KL divergence between $\hat{p}$ and D.

[7]This is in contrast to *event* level privacy, which would only hide the presence or absence of a single (or small number of) call records.

Example uses of synthetic Call Detail Records include estimating daily ranges (the maximum distance a person travels in one day), modeling epidemic routing, and the modeling of hypothetical cities, in which the analyst creates a paremterized model of a city and user behavior patterns that cannot be observed in the real world, yielding the power to experiment with the effects of modifications to reality such as telecommuting [23].

3.3 A Sketch of DP-WHERE

Each Call Detail Record corresponds to a single voice call or text message. Users making more than 120 calls in any hour are filtered out[8], and it is assumed that the number of remaining users, denoted n, is known. Each of the n users is identified by an integer in $\{1, \ldots, n\}$. The calls were made in a metropolitan area divided into smaller geographic areas according to a $d \times d$ grid. Each Call Detail Record is augmented with inferred home and work locations obtained by a combination of clustering and regression [22][9]. Thus, in DP-WHERE each element in the data set contains an id (number between 1 and n), date, time, latitude, longitude, and the inferred home and work locations.

The general approach is to create several distributions, all in a differentially private manner (Section 3.3.1). The synthetic Call Detail Records are generated by appropriate sampling from these distributions (Section 3.3.2).

3.3.1 DP-WHERE: Decription of the Distributions

Home and Work. For each of Home and Work, DP-WHERE computes a probability distribution on a square grid covering the metropolitan area in question, with a simple histogram query (Example 6 in Section 2.3 above). For example, for the Home distribution, the histogram reports, for each grid cell, the approximate (that is, noisy) number of users in the data set whose Home location is in this grid cell. In order to be able to transform this to a probability distribution – for example, to remove negative counts – post-processing techniques are applied that require no additional access to the true data [21]. The procedure is the same for differentially private release of the distribution on Work locations.

Commute Distance DP-WHERE uses a coarser grid for this computation, called the *commute grid*, obtained by merging neighboring cells in the original $d \times d$ grid. For each cell in the commute grid, DP-WHERE computes an empirical distribution on commute distances for people whose home location falls in this cell. This yields a cumulative distribution function on commute distances for each "home" grid cell.

We briefly describe the construction of one of these CDFs, say, for the ith grid cell. The algorithm creates a *data-dependent* histogram of commute distances for the residents in this cell. Each histogram bin is a range of distances, and the (true, non-noisy) count in bin j is the number of users living in the ith grid cell whose true commute distance is in the range associated with the jth bin. As above, the histogram is converted to a distribution via post-processing.

The subtlety is in determining the "right" set of bins for this histogram. This is done by assuming the commute distances for the residents of grid cell i are modeled by an exponential

[8]These are assumed to be auto-dialers.

[9]Since the clustering uses only information specific to the given user, together with global information about the locations of cell towers, the determination of these fields will not affect the privacy guarantee.

distribution of the form $\eta(x) = \lambda e^{-\lambda x}$ (each grid cell has its own distribution on commute distances, *i.e.*, its own λ). The *exponential mechanism* (Section 2.3) is used to approximate the median of this distribution: assuming the inputs $x_1, x_2, \ldots,$ are sorted, I_k is selected (but not revealed) with probability proportional to $(x_{k+1} - x_k)e^{|k-m|}$; the algorithm then selects an element uniformly from I_k and releases this value. Finally, λ is set at $\ln 2$ divided by this approximate median.

Once λ has been chosen, the histogram bins are defined according to the deciles of the exponential distribution with parameter λ.

Summarizing to this point, DP-WHERE finds, in a differentially private fashion, distributions on home locations and, independently, on work locations. For each cell in the commute grid, it finds a distribution on commute distances for residents in this cell. The techniques used include the Laplace mechanism for histogram queries, a post-processing algorithm (for turning noisy counts into a probability distribution), and the exponential mechanism (for selecting the median of a set of numbers, used to infer λ under the assumption that the distribution is exponential).

Calls per Day per User In this step, for a fixed, discrete, set of potential means $\mu \in \{\mu_1, \mu_2, ...\mu_m\}$ and standard deviations $\sigma \in \{\sigma_1, \sigma_2, \ldots, \sigma_s\}$, the algorithm computes a noisy count of the number of users whose (rounded) mean number of calls per day is μ with a (rounded) standard deviation of σ. Note that this is again a histogram query – the cells of the histogram correspond to (mean,deviation) pairs (μ_i, σ_j). Post-processing is again invoked to obtain a distribution.

2-Means Clustering. DP-WHERE runs a privacy-preserving 2-means clustering algorithm [2] to classify users based on a 24-dimensional probability vector describing their daily calling patterns. In a little more detail, consider a specific user i. Based on the Call Detail Records, DP-WHERE first constructs a *non-private* probability vector P_i for user i, using only the call records for user i, describing for each hour $j \in [24]$ the fraction of i's calls made in the jth hour of the day. These P_i's are not released. Instead, DP-WHERE clusters these 24-dimensional vectors into two clusters, using a differentially private algorithm. k-means clustering involves iteratively assigning an input to the kth nearest cluster center and then averaging the points assigned to each center in order to find the new center. They key points for doing this with privacy are (1) not revealing the assignment of points to centers and (2) privately carrying out the averaging by computing a noisy sum and dividing by a noisy count. In each iteration the algorithm applies post-processing to ensure that the new centers represent probability vectors, *i.e.*, that their entries are non-negative and sum to 1. The output at this step is a pair of cluster centers – probability vectors representing calling patterns – together with their approximate sizes. These sizes yield a distribution on the clusters.

Hourly Calls Per Location. The last set of probability distributions generated by DP-WHERE yield, for each hour of the day, a probability distribution over grid cells, intuitively describing where the population as a whole is likely to be during the given hour. That is, the Hourly Location distribution for hour $j \in [24]$ yields a probability distribution on locations (grid cells) for the population as a whole during the jth hour of every day covered by the data set. Ideally, this would be done by counting, for each grid cell and hour, the number of calls made from that grid cell during that hour of the day, summed over the different days covered by the data set. The difficulty is that these counts are highly sensitive, as only users making more than 120 calls in a single hour have been filtered out. For each hour, the total sensitivity of this computation is 120 times the number of days. Thus, even though, for a fixed $j \in [24]$, DP-WHERE builds something like a histogram, with one cell for each cell of the $d \times d$ geographic grid, the L_1-sensitivity of this data structure is 120 times the number of days. Applying the Laplace mechanism would add noise of this magnitude to *each* of the d^2 cells, which makes for too much distortion over all.

This difficulty is addressed using a *grouping and smoothing* technique [30], in which geographically close grid cells are "merged," essentially coarsening the geographic grid, to give a data structure with fewer elements, but the same L_1-sensitivity. Since the number of cells in the data structure is reduced, the total distortion is reduced, even though the distortion per (fat) cell remains unchanged. Now, each merged cell contains the (noisy) sum, S, of the number of calls during hour j in some number m of the original grid cells. DP-WHERE assigns to each of these m cells the value S/m. (The number m is fixed, public, and part of the algorithm.) Finally, as in all the other steps, post-processing is applied to convert the noisy counts to a probability distribution.

This completes the description of the (differentially privately generated distributions) used in generating synthetic Call Detail Records. No further access is made to the real data.

3.3.2 DP-WHERE: Generation of Synthetic CDRs

DP-WHERE, having found differentially private versions of the same distributions found by WHERE, creates records exactly as is done in WHERE. The first step is to generate synthetic users:

1. Sample from the (differentially privately generated) distribution on home locations, yiedling a geographic grid cell c;

2. Sample from the (differentially privately generated) distribution on commute distances for residents of geographic grid cell c, yielding a commute distance d.

3. Weight the cells at distance d according to their densities under the (differentially privately generated) Work distribution, normalize, and sample from the resulting probability distribution to obtain a work location.

4. Randomly assign the user a mean and standard deviation for the number of calls it will make per day, according to the (differentially privately generated) distribution on pairs (μ, σ) described in the paragraph above labeled **Calls Per Day Per User**.

5. Randomly assign the synthetic user to one of the two calling patterns learned in the differentially private 2-means clustering algorithm, according to the distribution on the clusters, released by that algorithm.

In the second step, the synthetic users are "moved" between their home and work locations. Fix $i \in \{1, 2, \ldots, n\}$. The procedure described next generates a day in the life of synthetic user i.

1. Generate a number N of calls to be made during the day by sampling from a normal distribution with mean μ and variance σ^2.

2. Allocate the total number N of calls to be made in this day to the 24 different hours of this day, according to the calling pattern (cluster) to which synthetic user i was assigned. Assign the exact time within the hour by interpolating between the beginning and end of the hour.

3. Finally, for each call made by user i during hour j, choose the location – select between user i's Home and Work location – by sampling according to the (differentially privately generated) hourly calls per location densities for these two locations during hour j.

location at

The generation of synthetic Call Detail Records relies only on differentially privately computed distributions. Not only does this mean that generation requires no further access to the original data, but also that the generated records provide no information beyond what is revealed by these distributions. Thus, the algorithm could release these differentially private distributions and allow the data analyst to generate synthetic data sets at will.

3.3.3 Exprimental Validation of DP-WHERE

Experiments were carried out using Call Detail Records from actual cellphone use over 91 consecutive days. The dataset contains over one billion records involving over 250,000 unique phones chosen at random from phones billed to ZIP codes within 50 miles of the center of New York City.

As in WHERE, accuracy of DP-WHERE is measured by a "normalized" Earth Mover Distance. The results vary according to the coarseness of the commute grid (in both WHERE and DP-WHERE) and the choice of the *total, cumulative* privacy loss (in DP-WHERE). For a commute grid cell size of 0.01^o, (non-private) WHERE yields an average hourly error of 3.2150; when $\varepsilon = 0.33$ (respectively, 0.23 and 0.13) this quantity is 3.5136 (respectively, 3.4066 and 5.3391). A coarser grid cell size of 0.05^o yields 3.0871 for WHERE and, respectively for these same values of ε, 4.5687, 5.1691, and 5.2754 for DP-WHERE. Even with 10,000 synthetic users moving across more than 14,000 square miles, the distance between the synthetic and real population density distributions for DP-WHERE differed by only between 0.17 and 2.2 miles from those of WHERE.

Experiments on (non-private) WHERE using public information, such as US census data, (and not Call Detail Records) were also carried out [23]. In all cases, DP-WHERE with Call Detail Records performed better than WHERE using public data. Thus, if the choice is between unfettered access to public data and differentially private access to the Call Detail Records, these experiments show that differential privacy, even with $\varepsilon = 0.13$, has better utility.

Experiments were also carried out to measure the daily range, or the maximum distance between any two points visited by an individual in a day. The boxplots for daily range in DP-WHERE ($\varepsilon = 0.23$), WHERE, and the real Call Detail Records are qualitatively similar, with differences of $0.5 - 1.3$ miles across the middle two quartiles (the smallest IQR of the three sets is 5.2 miles).

4 Challenges for Differential Privacy

The greatest *scientific* challenge is that, for a given computational task and a given value of ε, finding a low-error, differentially private algorithm can be hard. An analogy may be made to numerical analysis. Suppose, in the non-private world, we wish to compute a matrix decomposition. A naive algorithm for the decomposition may be numerically unstable, so we first consult a textbook on numerical algorithms and write our program based on the stable algorithm in the text. It is easy now – but quite possibly the algorithm in the text was a PhD thesis when it was developed in the 1970's.

A different sort of challenge is posed by "non-algorithmic" thinking in data analysis. From data cleaning through detailed investigation, many researchers who work with data do not, indeed can not, provide an algorithmic description of their interactions with the data. With no algorithm for the non-private case, there is essentially no hope of finding a differentially private alternative. This is less of an issue in machine learning and the VLDB (Very Large DataBases) communities, where the sheer volume of data rules out non-algorithmic approaches.

Differential privacy requires a new way of interacting with data, one that attempts to minimize privacy loss by reducing the explicit viewing, not to mention publication, of intermediate results. But *query minimization* is a completely foreign concept to data analysts. A good analogy might be to running an industrial scale database without the benefit of query planning, leading to (literally) prohibitive computational costs.

By far the hardest to grapple with are the *social* challenges of a changing world, in which highly detailed research data sets are expected to be shared and re-used, and all manner of information exploited for commercial gain, seemingly without limit. That this is fundamentally incompatible with privacy is proved by a host of lower bounds and attacks[10]. What are we to make of this state of affairs? To paraphrase Latanya Sweeney [35], computer science got us into this mess, can computer science get us out of it?

One thing seems certain: complexity of this type requires a mathematically rigorous theory of privacy and its loss. Other fields – economics, ethics, policy – cannot be brought to bear without a "currency," or measure of privacy, with which to work. In this connected world, we cannot discuss tradeoffs between privacy and statistical utility without a measure that captures cumulative harm over multiple releases.

Publish the Loss, and Pay a Fine for Infinity. Whatever the measure of privacy loss on which the community ultimately settles, we should take a page from environmental law and require measurements of privacy loss to be made public. In differential privacy, simply ensuring that the loss is finite helps to protect against many common avenues of attack. Although the field of differentially private algorithms has many exciting results, provided by researchers from an increasingly broad array of disciplines, much remains to be done. Perhaps publication of privacy losses will lead to competition, deploying the talents of an even larger set of researchers and other marketers and consumers of data in the search for private algorithms.

References

[1] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs# 1. In *CRYPTO*, pages 1–12, 1998.

[2] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. In *Proc. 24th ACM Symposium on Principles of Database Systems*, pages 128–138, 2005.

[3] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proc. 40th ACM SIGACT Symposium on Thoery of Computing*, pages 609–618, 2008.

[4] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography (TCC)*, pages 253–273. 2011.

[5] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proc. 22nd ACM Symposium on Principles of Database Systems*, pages 202–210, 2003.

[10]It is also fundamentally incompatible with statistical power, where issues of false discovery arise.

[6] C. Dwork. Differential privacy. In *Proc. 33rd International Colloquium on Automata, Languages and Programming (ICALP)(2)*, pages 1–12, 2006.

[7] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: privacy via distributed noise generation. In *Advances in Cryptology: Proceedings of EUROCRYPT*, pages 486–503, 2006.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. 3rd Theory of Cryptography Conference*, pages 265–284, 2006.

[9] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of lp decoding. In *Proc. 39th ACM Symposium on Theory of Computing*, pages pp. 85–94, 2007.

[10] C. Dwork, M. Naor, O. Reingold, G. Rothblum, and S. Vadhan. When and how can privacy-preserving data release be done efficiently? In *Proc. 41st ACM Symposium on Theory of Computing*, pages 381–390, 2009.

[11] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Advances in Cryptology – CRYPTO'04*, pages 528–544, 2004.

[12] C. Dwork and S. Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *Proceedings of CRYPTO 2008*, pages 468–480, 2008.

[13] Cynthia Dwork and Moni Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2010.

[14] Cynthia Dwork and Guy Rothblum. Concentrated differential privacy, 2013. manuscript.

[15] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60, 2010.

[16] Stephen Fienberg, Alessandro Rinaldo, and Xiaolin Yang. Differential privacy and the risk-utility tradeoff for multi-dimensional contingency tables. In *Privacy in Statistical Databases*, pages 187–199, 2011.

[17] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jonathan Ullman. Privately releasing conjunctions and the statistical query barrier. In *STOC '11*, pages 803–812, 2011.

[18] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems 25*, pages 2348–2356, 2012.

[19] M. Hardt and K. Talwar. On the geometry of differential privacy. In *Proc. 42nd ACM Symposium on Theory of Computing*, pages 705–714, 2010.

[20] Moritz Hardt and Guy Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 61–70, 2010.

[21] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1-2):1021–1032, 2010.

[22] Sibren Isaacman, Richard Becker, Ramón Cáceres, Stephen Kobourov, Margaret Martonosi, James Rowland, and Alexancer Varshavsky. Identifying important places in peoples lives from cellular network data. In *Pervasive Computing*, pages 133–151. 2011.

[23] Sibren Isaacman, Richard Becker, Ramón Cáceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. Human mobility modeling at metropolitan scales. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 239–252, 2012.

[24] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 277–286, 2008.

[25] F. McSherry. Privacy integrated queries (codebase). available on Microsoft Research downloads website. See also pages 19-30, Proc. SIGMOD 2009.

[26] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proc. 48th Annual Symposium on Foundations of Computer Science*, 2007.

[27] Darakhshan Mir, Sibren Isaacman, Ramón Cáceres, margaret Martonosi, and Rebecca WrightN. Dp-where: Differentially private modeling of human mobility. In *IEEE BigData*, 2013.

[28] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 111–125, 2008.

[29] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. *STOC*, 2013.

[30] S. Papadopoulos and G. Kellaris. Practical differential privacy via grouping and smoothing. In *Proceedings of the 39th international conference on Very Large Data Bases*, pages 301–312, 2013.

[31] Manoj Prabhakaran and Amit Sahai. *Secure Multi-party Computation.* IOS Press, 2013.

[32] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 765–774, 2010.

[33] Indrajit Roy, Srinath Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and privacy for mapreduce. In *NSDI*, volume 10, pages 297–312, 2010.

[34] L. Sweeney. Weaving technology and policy together to maintain confidentiality. *J. Law Med. Ethics*, 25:98–110, 1997.

[35] Latanya Sweeney, 2012. Keynote Lecture, Second Annual iDASH All-Hands Symposium.

[36] Jonathan Ullman and Salil P. Vadhan. PCPs and the hardness of generating private synthetic data. In *TCC '11*, pages 400–416, 2011.

[37] S. Warner. Randomized response: a survey technique for eliminating evasive answer bias. *JASA*, pages 63–69, 1965.