1

Algorithmic Privacy

From Anonymity to Noise

"ANONYMIZED DATA ISN'T"

It has been difficult for medical research to reap the fruits of large-scale data science because the relevant data is often highly sensitive individual patient records, which cannot be freely shared. In the mid-1990s, a government agency in Massachusetts called the Group Insurance Commission (GIC) decided to help academic researchers by releasing data summarizing hospital visits for every state employee. To keep the records anonymous, the GIC removed explicit patient identifiers, such as names, addresses, and social security numbers. But each hospital record did include the patient's zip code, birthdate, and sex—these were viewed as useful summary statistics that were coarse enough that they could not be mapped to specific individuals. William Weld, the governor of Massachusetts, assured voters

that the removal of explicit patient identifiers would protect patient privacy.

Latanya Sweeney, who was a PhD student at MIT at the time, was skeptical. To make her point, she set out to find William Weld's medical records from the "anonymous" data release. She spent $20 to purchase the voter rolls for the city of Cambridge, Massachusetts, where she knew that the governor lived. This dataset contained (among other things) the name, address, zip code, birthdate, and sex of every Cambridge voter—including William Weld's. Once she had this information, the rest was easy. As it turned out, only six people in Cambridge shared the governor's birthday. Of these six, three were men. And of these three, only one lived in the governor's zip code. So the "anonymized" record corresponding to William Weld's combination of birthdate, sex, and zip code was unique: Sweeney had identified the governor's medical records. She sent them to his office.

In retrospect, the problem was that although birthdate, sex, and zip code could not be used individually to identify particular individuals, in combination they could. In fact, Sweeney subsequently estimated from US Census data that 87 percent of the US population can be uniquely identified from this data triple. But now that we know this, can the problem of privacy be solved by simply concealing information about birthdate, sex, and zip code in future data releases?

It turns out that lots of less obvious things can also identify you—like the movies you watch. In 2006, Netflix launched the Netflix Prize competition, a public data science competition to find the best "collaborative filtering" algorithm to power Netflix's movie recommendation engine. A key feature of Netflix's service is its ability to recommend to users movies that they might like, given how they have rated past movies. (This was especially important when Netflix was primarily a mail-order DVD rental service, rather than a streaming service—it was harder to quickly browse or sample movies.) Collaborative filtering is a kind of machine learning problem designed to recommend

purchases to users based on what similar users rated well. For every user, Netflix had the list of movies that person had rated. For each movie, Netflix knew both what score the user had given the movie (on a scale of 1 to 5 stars) and the date on which the user provided the rating. The goal of a collaborative filtering engine is to predict how a given user will rate a movie she hasn't seen yet. The engine can then recommend to a user the movies that it predicts she will rate the highest.

Netflix had a basic recommendation system based on collaborative filtering, but the company wanted a better one. The Netflix Prize competition offered $1 million for improving the accuracy of Netflix's existing system by 10 percent. A 10 percent improvement is hard, so Netflix expected a multiyear competition. An improvement of 1 percent over the previous year's state of the art qualified a competitor for an annual $50,000 progress prize, which would go to the best recommendation system submitted that year. Of course, to build a recommendation system, you need data, so Netflix publicly released a lot of it—a dataset consisting of more than a hundred million movie rating records, corresponding to the ratings that roughly half a million users gave to a total of nearly eighteen thousand movies.

Netflix was cognizant of privacy concerns: it turns out that in the United States, movie rental records are subject to surprisingly tough privacy laws. The Video Privacy Protection Act was passed by the United States Congress in 1988, after Robert Bork's video rentals were published in the *Washington City Paper* during his Supreme Court nomination hearings. The law holds any video rental provider liable for up to $2,500 in damages per customer whose records are released. So, just as the state of Massachusetts had done, Netflix removed all user identifiers. Each user was instead represented with a unique but meaningless numeric ID. This time there was no demographic information at all: no gender, no zip code. The only data about each user was his or her movie ratings.

Yet only two weeks after the data release, Arvind Narayanan (who was a PhD student at the University of Texas at Austin) and his advisor, Vitaly Shmatikov, announced that they could attach real names to many of the "anonymized" Netflix records. In their research paper, they wrote:

> We demonstrate that an adversary who knows only a little bit about an individual subscriber can easily identify his or her record if it is present in the dataset, or, at the very least, identify a small set of records which include the subscriber's record. The adversary's background knowledge need not be precise, e.g., the dates may only be known to the adversary with a 14-day error, the ratings may be known only approximately, and some of the ratings and dates may even be completely wrong.

Regarding the Netflix dataset, they found that if an attacker knew the approximate dates (give or take a couple of weeks) when a target had rated six movies, he could uniquely identify that person 99 percent of the time. They also showed that this could be done at a large scale, by cross-referencing the Netflix dataset with Internet Movie Database (IMDB) movie ratings—which people post publicly under their own names.

But if people are posting about which movies they watch publicly, can identifying them in the Netflix dataset really be considered a privacy violation? Yes. People may publicly review only a small fraction of the movies they watch, but rate all of them on Netflix. The public reviews may suffice to reveal their Netflix identity, which then unveils all the movies they have watched and rated—which can expose sensitive information, including political leanings and sexual orientation. In fact, a gay mother of two who was not open about her sexual orientation sued Netflix, alleging that the ability to de-anonymize the dataset "would negatively affect her ability to pursue her livelihood and

support her family, and would hinder her and her children's ability to live peaceful lives." She was worried that her sexual orientation would become clear if people knew what movies she watched on Netflix. The lawsuit sought the maximum penalty allowed by the Video Privacy Protection Act: $2,500 for each of Netflix's more than two million subscribers. Netflix settled this lawsuit for undisclosed financial terms, and cancelled a planned second Netflix Prize competition.

The history of data anonymization is littered with many more such failures. The problem is that a surprisingly small number of apparently idiosyncratic facts about you—like when you watched a particular movie, or the last handful of items you purchased on Amazon—are enough to uniquely identify you among the billions of people in the world, or at least among those appearing in a large database. When a data administrator is considering releasing an "anonymized" data source, he can try to make an informed guess about how difficult it would be for an attacker to reidentify individuals in the dataset. But it can be difficult for him to anticipate other data sources—like the IMDB reviews—that might be available. And once data is released on the Internet, it cannot be revoked in any practical sense. So the data administrator must be able to anticipate not just every attack that can be launched using other data sources that are currently available but also attacks that might use data sources that become available in the future. This is essentially an impossible task, and the reason that Cynthia Dwork (one of the inventors of differential privacy, which we will discuss later in this chapter) likes to say that "anonymized data isn't"—either it isn't really anonymous or so much of it has been removed that it is no longer data.

A BAD SOLUTION

How can we fix the problem of de-anonymization, also known as re-identification? In both the Massachusetts hospital records and Netflix

examples, the trouble was that there were lots of unique records in the dataset. Only one male Cambridge resident living in the 02139 zip code was born on December 18, 1951; only one Netflix user had watched *Hour of the Wolf*, *Brazil*, *Matinee*, and *The City of Lost Children* in March 2005. The problem with unique records is that they are akin to a fingerprint that can be reconnected with an identity by anyone who knows enough about someone to identify that person's record—and then she can learn everything else contained in that record.

For example, consider the following fictional table of patients at the Hospital of the University of Pennsylvania (HUP). Even with names removed, anyone who knows that Rebecca is a patient at HUP who also knows her age and gender can learn that she has HIV, because those attributes uniquely identify her within this database.

Name	Age	Gender	Zip Code	Smoker	Diagnosis
Richard	64	Male	19146	Y	Heart disease
Susan	61	Female	19118	N	Arthritis
Matthew	67	Male	19104	Y	Lung cancer
Alice	63	Female	19146	N	Crohn's disease
Thomas	69	Male	19115	Y	Lung cancer
Rebecca	***56***	***Female***	19103	N	HIV
Tony	52	Male	19146	Y	Lyme disease
Mohammed	59	Male	19130	Y	Seasonal allergies
Lisa	55	Female	19146	N	Ulcerative colitis

Hypothetical database of patient records, in which there is only one 56-year-old female, thus enabling anyone who knows Rebecca and the fact that she is a patient to infer that she has HIV.

An initial idea for a solution, called *k*-anonymity, is to redact information from individual records so that no set of characteristics matches just a single data record. Individual characteristics are divided into "sensitive" and "insensitive" attributes. In our fictional table, the diagnosis is sensitive, and everything else is insensitive. The goal of

k-anonymity is to make it hard to link insensitive attributes to sensitive attributes. Informally, a released set of records is *k*-anonymous if any combination of insensitive attributes appearing in the database matches at least *k* individuals in the released data. There are two main ways to redact information in a table to make it *k*-anonymous: we can suppress information entirely (that is, not include it at all in the released data), or we can coarsen it (not release the information as precisely as we know it, but instead bucket it). Consider the following redacted table of our fictional HUP patients:

Name	Age	Gender	Zip Code	Smoker	Diagnosis
*	60–70	Male	191**	Y	Heart disease
*	60–70	Female	191**	N	Arthritis
*	60–70	Male	191**	Y	Lung cancer
*	60–70	Female	191**	N	Crohn's disease
*	60–70	Male	191**	Y	Lung cancer
*	***50–60***	***Female***	191**	N	HIV
*	50–60	Male	191**	Y	Lyme disease
*	50–60	Male	191**	Y	Seasonal allergies
*	***50–60***	***Female***	191**	N	Ulcerative colitis

The same database after 2-anonymous redactions and coarsening. Now two records match Rebecca's age range and gender.

The table has been modified: names have been redacted, and ages and zip codes have been "coarsened" (ages are now reported as ten-year ranges, and zip codes are reported only to the first three digits). The result is that the table is now 2-anonymous. Any insensitive information we might know about a person—for example, that Rebecca is a fifty-six-year old female—now corresponds to at least two different records. So no person's record can be uniquely reidentified from his or her insensitive information.

Unfortunately, although *k*-anonymity can prevent record reidentification in the strictest sense, it highlights that reidentification is not

the only (or even the main) privacy risk. For example, suppose we know that Richard is a sixty-something male smoker who is a patient at HUP. We can't identify his record—this information corresponds to three records in the modified table. But two of those records correspond to patients with lung cancer, and one corresponds to a patient with heart disease—so we can be sure that Richard has either lung cancer or heart disease. This is a serious privacy violation, and not one that is prevented by *k*-anonymity.

But the concept of *k*-anonymity also suffers from an even worse and more subtle problem, which is that its guarantees go away entirely when multiple datasets are released, even if all of them have been released with a *k*-anonymity guarantee. Suppose that in addition to knowing that Rebecca is a fifty-six-year-old female nonsmoker, we know that she has been a patient at two hospitals: HUP and the nearby Pennsylvania Hospital. And suppose that both hospitals have released *k*-anonymous patient records. HUP released the 2-anonymous records we saw above, and Pennsylvania Hospital released the following 3-anonymous table:

Name	Age	Gender	Zip Code	Diagnosis
*	***50–60***	***Female***	191**	HIV
*	***50–60***	***Female***	191**	Lupus
*	***50–60***	***Female***	191**	Hip fracture
*	60–70	Male	191**	Pancreatic cancer
*	60–70	Male	191**	Ulcerative colitis
*	60–70	Male	191**	Flu-like symptoms

In this 3-anonymous database from a different hospital in which Rebecca is a patient, three records match her age and gender. By combining this database with the 2-anonymous one from HUP, we can again unambiguously infer that Rebecca has HIV.

Individually, the tables satisfy *k*-anonymity, but together, there is a problem. From the first table, we can learn that Rebecca has either HIV or ulcerative colitis. From the second table, we learn that Rebecca has either HIV, lupus, or a hip fracture. But when the tables are taken

together, we know with certainty that she has HIV. So there are two major flaws with *k*-anonymity: it tries to protect against only a very narrow view of what a privacy violation is—an explicit reidentification of patient records—and its guarantees are brittle under multiple data releases.

Another tempting first thought, if our goal is to prevent reidentification, is that the solution to private data analysis is simply not to release any data at the individual level. We should restrict ourselves to releasing only aggregate data—averages over many people, say, or predictive models derived using machine learning. That way, there is nothing to reidentify. But this turns out to be both overly restrictive and again insufficient. Aggregating data turns out to pose plenty of privacy risks, even as it limits the utility of the data. Consider the following example, which shocked the genetics community when it was first discovered in 2008.

The human genome is encoded via a sequence of roughly three billion base pairs—complementary pairs of nucleobases that form the basic building blocks of DNA. Any two people will have genomes that are identical in almost all positions—more than 99 percent of them. But there are certain positions in the genome that may differ between two individuals. The most common forms of genetic variation are called single nucleotide polymorphisms, or SNPs (pronounced "snips"). A SNP represents a position in the genome in which one person might have a certain base pair but another person might have a different one. There are roughly ten million SNPs in the human genome. SNPs can be useful in identifying the genetic causes of disease. The goal of a genome-wide association study (GWAS) is often to find correlations between the presence of genetic variants (alleles) in SNPs and the prevalence of a disease. Generating a basic form of GWAS data might involve gathering a thousand patients who all have some disease—say, pancreatic cancer—sequencing their DNA, and publishing the average allele frequency in each SNP across the entire population. Note that this form of data consists only of averages or statistics: for example,

that in a certain SNP position, 65 percent of the population have the C nucleotide and 35 percent have the A nucleotide instead. But because there are so many SNPs, there can be lots of these averages in a single dataset—hundreds of thousands or millions.

The 2008 paper showed that by combining a huge number of simple tests for correlation (one for each SNP), it was possible to test whether a *particular individual's* DNA had been used in computing the average allele frequencies in a particular GWAS dataset—or in other words, whether a particular individual was part of the group of people from whom the GWAS data was gathered. This is not a reidentification, as only one bit of information has been learned: whether the individual was in the group or not. But this is nevertheless a significant privacy concern, because pools of subjects from which GWAS data is gathered often share some disease trait. Learning that an individual was in the GWAS group can reveal that he has, say, pancreatic cancer.

In response to this study, the National Institutes of Health immediately removed all aggregate GWAS data from open-access databases. This mitigated the privacy problem but erected a serious barrier to scientific research of societal importance. More recent research has shown that many other aggregate statistics can leak private data. For example, given only input-output access to a trained machine learning model, it is often possible to identify the data points that were used in its training set. Roughly speaking, this is because machine learning models will at least slightly "overfit" the particular examples they were trained on—for example, a trained model will have higher confidence when asked to classify photos that were used to train it, compared to the confidence it will have when asked to classify examples it has not previously seen.

BREACHES AND INFERENCES

In general, notions of data privacy based on anonymization are susceptible to the serious flaws we have outlined. It might be tempting to

instead think that the powerful toolkit of modern cryptography could play an important role in privacy. But it turns out that cryptography solves a different problem, which we might more accurately call data security, rather than providing the type of privacy we're looking for. It's instructive to understand why.

Imagine a simple computational pipeline for medical research. It begins with a database of patient medical records, and our goal is to use machine learning to build a model predicting whether patients have a particular disease based on their observed symptoms, test results and medical history. So the database is given as input to a machine learning algorithm like backpropagation, which in turn outputs the desired predictive neural network.

In this pipeline, we of course want the raw database to be secured. Only authorized personnel—doctors and the researchers building the predictive model—should be allowed to read or alter the database. We want to prevent outright *breaches* of the data (such as the major ones that have occurred at companies like Equifax, Marriott, and Yahoo in recent years). This is the core privacy problem that cryptography attempts to solve, in the form of file encryption algorithms. A useful metaphor is a lock and keys—when the database is locked, only those with the keys should be able to unlock it.

The neural network, however, is different from the original data. We *want* it to be published, released and used—not only by the authorized doctors and researchers, but really by almost anyone. A natural outcome would be for the researchers to publish the details of their model in a scientific journal, so that other researchers and doctors can understand the interactions between various symptoms and the disease in question. Encrypting the neural network, or preventing its use in the field, would obviate the purpose of building it in the first place. In general the purpose of doing computations on data, even including sensitive private data, is to release at least some broad properties about that data to the world at large.

While we are not concerned with unauthorized breaches of the neural network (releasing it was the whole point), we should be concerned about unwanted *inferences* that might be made from it. In particular, we wouldn't want the release of the neural network to allow someone to determine specific details about your medical record, such as whether you have the disease in question. (In fact, recent research has discovered that it often is possible to do exactly that—extract the training data just from access to the learned model.) This is a more nuanced concern than simply locking data down—we want the results of our algorithms to release *useful* information, but not to leak *private* information.

This is the notion of privacy we are concerned with here. And its importance has been rapidly amplified by the era we live in, in which sensitive consumer data is used to build predictive models that are then "released," in the sense that they are used by a very large and diffuse set of entities such as apps, employers, advertisers, lenders, insurers, judges and others. While the design of cryptographic algorithms has been studied for centuries (and experienced colossal scientific and practical advances beginning in the 1970s, with the introduction of so-called public key cryptography), the more subtle inferential privacy we seek is in its relative infancy.

So there are privacy risks beyond simple reidentification. Limiting ourselves to aggregate statistics doesn't fix the problem. And privacy (at least of the type we are seeking) is not the problem that cryptography solves. One way to start thinking rigorously about what to do about all this is to first consider what risks we want to mitigate with data privatization. Perhaps we can start by being ambitious. Can we ask that performing a data analysis should have no risks at all for the people whose data is involved in the analysis? If we could do this, it would seem to be a great privacy definition—the ability to promise complete freedom from harm. We will ultimately see how we can usefully achieve something that is almost as strong—but the following

thought experiment illustrates that we will have to refine our goals a bit, and limit our ambitions.

SMOKING MAY BE HARMFUL TO YOUR PRIVACY

Imagine a man named Roger—a physician working in London in 1950 who is also a cigarette smoker. This is before the British Doctors Study, which provided convincing statistical proof linking tobacco smoking to increased risk for lung cancer. In 1951, Richard Doll and Austin Bradford Hill wrote to all registered physicians in the United Kingdom and asked them to participate in a survey about their physical health and smoking habits. Two-thirds of the doctors participated. Although the study would follow them for decades, by 1956 Doll and Hill had already published strong evidence linking smoking to lung cancer. Anyone who was following this work and knows Roger would now increase her estimate of Roger's risk for lung cancer simply because she knows both that he is a smoker and now a relevant fact about the world: that smokers are at increased risk for lung cancer. This inference might lead to real harm for Roger. For example, in the United States, it might cause him to have to pay higher health insurance premiums—a precisely quantifiable cost.

In this scenario, in which Roger comes to harm as the direct result of a data analysis, should we conclude that his privacy was violated by the British Doctors Study? Consider that the above story plays out in exactly the same way even if Roger was among the third of British physicians who declined to participate in the survey and provide their data. The effect of smoking on lung cancer is real and can be discovered with or without any particular individual's private data. In other words, the harm that befell Roger was not because of something that someone identified about his data per se but rather because of a general fact about the world that was revealed by the study. If we were to call this a privacy violation, then it would not be possible to conduct

any kind of data analysis while respecting privacy—or, indeed, to conduct science, or even observe the world around us. This is because any fact or correlation we observe to hold in the world at large may change our beliefs about an individual if we can observe one of the variables involved in the correlation.

How, then, should we refine our goals so as to distinguish between protecting the secrets specific to Roger's data while still allowing the discovery of facts about the world that might nevertheless cause people to think differently about Roger? Our first attempt, which unhelpfully would have declared the British Doctors Study a violation of Roger's privacy, can be viewed as comparing the world as it actually is with the imaginary world in which the British Doctors Study was never carried out. It declares Roger's privacy to have been violated because Roger was demonstrably better off in the imaginary world, as compared to the real world. This view of privacy is consistent with what Tore Dalenius defined in 1977 as a goal for statistical database privacy: that nothing about an individual should be learnable from a dataset if it cannot also be learned without access to the dataset. But this thought experiment didn't get to the heart of what we want privacy to mean here—a measure of the harm that might befall Roger *as the result of the use of his data*. To capture this nuance, we can consider a slightly different thought experiment.

Suppose we compare the following two worlds. In the first world, the British Doctors Study is carried out, and Roger opts in to it, so the study includes his data. In the second world, the British Doctors Study is still carried out, but this time Roger opts out of it—so in this world, his data has no effect on the outcome. In both cases, the participation of everyone other than Roger is fixed; the only difference between the two worlds is whether Roger's data was used in the study or not. What if we declare the study to be privacy-preserving if we can promise that Roger is no worse off if the study is performed with his data, compared to if it is performed without his data? In other words,

we ask for a refinement of Dalenius's goal: that nothing about an individual should be learnable from a dataset that cannot be learned from the *same dataset but with that individual's data removed.* This kind of definition still has the promise that Roger cannot be harmed by the use of his data, and it might plausibly allow useful science such as the British Doctors Survey to be carried out. In that scenario, Roger's insurance rates were raised because of the discovered link between smoking and lung cancer—but that fact about the world would have been discovered whether or not he opted into the study.

A DIFFERENT(IAL) NOTION OF PRIVACY

Enter differential privacy, a stringent measure of privacy that still allows us to gain useful insights from data. Differential privacy is a mathematical formalization of the foregoing idea—that we should be comparing what someone might learn from an analysis if any particular person's data was included in the dataset with what someone might learn if it was not. The concept was developed in the early 2000s by a team of theoretical computer scientists who were subsequently awarded the prestigious Gödel Prize: Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. To describe what differential privacy asks for, it is important to first understand what a randomized algorithm is.

Let's start with an algorithm. Remember, this is just a precise description of how to take inputs to some problem and process them to arrive at the desired outputs. In the introduction, we saw an example of a sorting algorithm. But an algorithm might also take as input health records and map them to a set of features that appear to be correlated. Or it might take as input video rental records and map them to a set of movie recommendations for each customer. The important thing is that the algorithm precisely specifies the process by which the inputs are mapped to the outputs. A randomized algorithm is just

an algorithm that is allowed to use randomization. Think about it as an algorithm that can flip coins as part of its precisely specified process and then make decisions that depend on the outcome of the coin flips. So a randomized algorithm maps inputs to the probabilities of different outputs. (We'll see a concrete example of a simple randomized algorithm in the next section.)

The introduction of randomness into algorithms has varied and powerful uses, including for the generation of cryptographic keys, for speeding up the search for solutions of algebraic equations, and for balancing out the loads on a distributed collection of servers. The use of randomness in differential privacy is for yet another purpose–namely, to deliberately add *noise* to computations, in a way that promises that any one person's data cannot be reverse-engineered from the results.

Differential privacy requires that adding or removing the data record of a single individual not change the probability of any outcome by "much" (we'll explain what this means in a minute). It requires this of an algorithm even in the worst case, no matter what records the other individuals have provided and no matter how unusual the added or removed data is. And differential privacy is a constraint that comes with a tunable "knob" or parameter, which can be thought of as a measure of the amount of privacy demanded. This knob governs how much a single individual's data can change the probability of any outcome. For example, if the privacy knob is set to 2.0, then differential privacy requires that no outcome be more than twice as likely if the algorithm is run with Roger's data compared with the case in which it is run on the same dataset but with Roger's data removed.[1]

Let's consider for a moment why the mathematical constraint of differential privacy corresponds to something we might think of as a

[1] We are actually describing the exponential of the privacy parameter, as usually defined in the mathematical literature. If we were to be consistent with the mathematical literature, we would say that the privacy parameter was ln(2) if changing Roger's data could cause an event to double in probability.

notion of privacy. We'll give three interpretations here, but there are more.

Most basically, differential privacy promises safety against arbitrary harms. It guarantees that no matter what your data is, and no matter what thing you are concerned about occurring because of the use of your data, that thing becomes (almost) no more likely if you allow your data to be included in the study, compared to if you do not. It literally promises this about anything you can think of. It promises that the probability that you get annoying telemarketing calls during dinner does not increase by very much if you allow your data to be included in a study. It promises that the probability that your health insurance rates go up does not increase by very much if you allow your data to be included in a study. And it certainly promises that the probability that your data record is reidentified (as in the Massachusetts hospital record and Netflix Prize examples) does not increase by very much. The strength of the promise of differential privacy is that it does not matter what harm you are worried about—differential privacy promises that the risk of *any* harm does not increase by more than a little bit as the result of the use of any individual's data. And it makes sense even in settings in which there is no data to reidentify, as in the GWAS (genomics) example.

But what if privacy is about what others can learn about you, instead of what bad things can happen to you? Differential privacy can also be interpreted as a promise that no outside observer can learn very much about any individual because of that person's specific data, while still allowing observers to change their beliefs about particular individuals as a result of learning general facts about the world, such as that smoking and lung cancer are correlated.

To clarify this, we need to think for a moment about how learning (machine or otherwise) works. The framework of Bayesian statistics provides a mathematical formalization of learning. A learner starts out with some set of initial beliefs about the world. Whenever he

observes something, he changes his beliefs about the world. After he updates his beliefs, he now has a new set of beliefs about the world (his posterior beliefs). Differential privacy provides the following guarantee: for every individual in the dataset, and for any observer no matter what their initial beliefs about the world were, after observing the output of a differentially private computation, their posterior belief about anything is close to what it would have been had they observed the output of the same computation run without the individual's data. Again, "close" here is governed by the privacy parameter or knob.

As one final way of interpreting this same privacy guarantee, suppose some outside observer is trying to guess whether a particular person—say, Rebecca—is in the dataset of interest or not (or whether her record specifies some particular ailment, such as lung cancer, or not). The observer is allowed to use an arbitrary rule to make his guess, based on the output of the differentially private computation. If the observer is shown either the output of a computation run with Rebecca's data or the output of the same computation run without her data, he will not be able to guess which output he was shown substantially more accurately than random guessing.

The three interpretations provided above are really just three different ways of looking at the same guarantee. But we hope they will convince you of what they have convinced many scientists: that differential privacy is among the strongest kinds of individual privacy assurances we could hope to provide without a wholesale ban on any practical use of data (for example, never doing valuable studies like the British Doctors Study linking smoking and lung cancer). The main question is whether it might be *too* strong, presenting such an onerous constraint that, for example, it is incompatible with modern machine learning. As we shall see, there is a rather happy middle ground between differential privacy and machine learning (and many other types of computations).

HOW TO CONDUCT EMBARRASSING POLLS

Differential privacy promises individuals a very strong kind of protection. But it would not be interesting if it were impossible to achieve while doing any kind of useful data analysis. Fortunately, this is not the case. In fact, it is possible to conduct essentially any kind of statistical analysis subject to differential privacy. But privacy doesn't come for free: one will generally need more data to obtain the same level of accuracy than one would need without a privacy constraint, and the more stringently one sets the privacy parameter, the more serious this tradeoff becomes.

To see an example of why this is possible, let's consider the simplest of all possible statistical analyses: computing an average. Suppose we want to conduct a poll to find out how many men in Philadelphia have ever cheated on their wives. A straightforward way to do this would be to attempt to randomly sample a reasonable number of men from the population of Philadelphians, call them, and ask them if they have ever had an affair. We would write down the answer provided by each one. Once we collected all of the data, we would enter it into a spreadsheet and compute the average of the responses, and perhaps some accompanying statistics (such as confidence intervals or error bars). Note that although all we wanted was a statistic about the population, we would have incidentally collected lots of compromising information about specific individuals. Our data could be stolen or subpoenaed for use in divorce proceedings, and as a result, people might rightly be concerned about participating in our poll. Or they might simply be embarrassed to admit to having had an affair, even to a stranger.

But consider the following alternative way of conducting a poll. Again, we would randomly sample a reasonable number of men from the population of Philadelphians (somewhat more than we needed before). We would call them up and ask them if they have ever had an

affair. But rather than asking them to answer our question directly, we would give them the following instructions: Flip a coin, and don't tell us how it landed. If it came up heads, tell us (honestly) whether you have ever cheated on your wife. But if it came up tails, tell us a *random* answer: flip the coin again and tell us yes if it comes up heads and no if it comes up tails. This polling protocol is an example of a simple randomized algorithm.

The result is that when we ask people whether they have ever had an affair, three-quarters of the time they tell us the truth (half the time the protocol tells them to tell us the truth, and if instead the protocol tells them to respond with a random answer, then half of *that* time they just happen to tell us the right answer at random). The remaining one-quarter of the time they tell us a lie. We have no way of telling true answers from lies. We record the answers that they give us—but now everyone has a strong form of plausible deniability. If in a divorce proceeding our records are subpoenaed and it is revealed that a man in our survey answered yes to our survey protocol, the accused husband can reasonably protest that in fact he had never had an affair—the coins simply came up such that he was asked to randomly report yes. Indeed, nobody can form strong beliefs about the true data of any single individual when records are collected in this way.

Yet from data collected in this way, it is still possible to get a highly accurate estimate of the fraction of Philadelphia men who have cheated on their wives. The key is that although the individual answers we are collecting are now highly error-prone, we know exactly how those errors are introduced, and so we can work backward to remove them, at least in the aggregate. Suppose, for example, that it happens to be that one-third of men in Philadelphia have at some point cheated on their wives. Then how many people in our survey do we expect to answer yes? We can compute this, since we know that people answer truthfully three-quarters of the time. One-third of the people in the

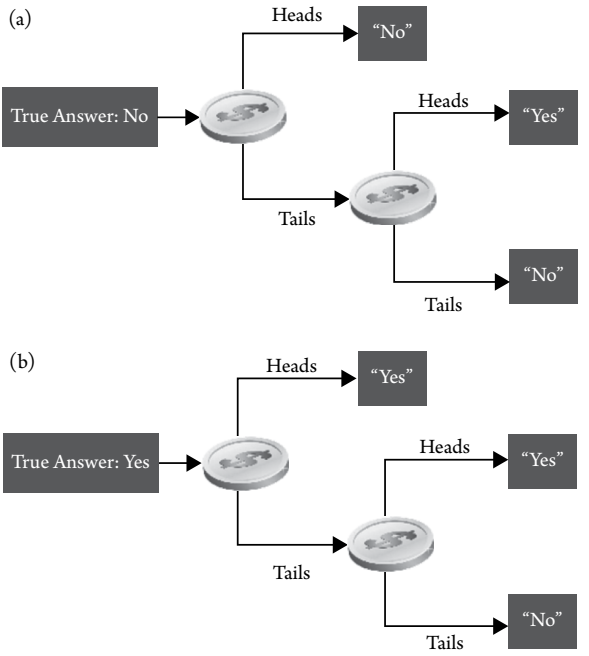

Fig. 3. If your true answer is no, the randomized response answers no two out of three times. It answers no only one out of three times if your answer is yes.

survey have a truthful answer of yes, and three-quarters of those will report yes, for a total of $1/3 \times 3/4 = 1/4$ of the population. Additionally, two-thirds of the people in the survey have a truthful answer of no, and one-quarter of these will report yes, for a total of $2/3 \times 1/4 = 1/6$. In total, we expect $1/4 + 1/6 = 5/12$ of the population to report yes in this case.

Because we know this, if we observe that five-twelfths of our survey population have reported yes, we can work backward to deduce that approximately one-third of male Philadelphians have cheated on their wives. This holds for any other proportion we observe—since we know the process by which errors have been introduced, we can work backward to deduce approximately the fraction of the population for whom the truthful answer is yes. This process is approximate, because even if exactly one-third of the men we survey have cheated on their

wives, this only tells us that five-twelfths of the population will report yes *on average*. The actual fraction will deviate slightly from five-twelfths because of how individuals' coin flips actually turn out. But because we are now reasoning about an average computed over a large sample, it will be very accurate—and will get more accurate the more people we survey. This is exactly the same effect we see when we flip coins. If we flip ten coins, we might expect the fraction of heads we see in total to deviate substantially from one-half. But if we flip ten thousand coins, we expect the fraction of heads that we see to be very close to one-half indeed. In exactly the same way, the error introduced in our survey by the randomness that was added for privacy shrinks to zero as we include more and more people in our survey. This is just an instance of what is known as the "law of large numbers" in statistics.

Although this protocol is simple, the result is remarkable: we are able to learn what we wanted without incidentally collecting any strongly incriminating information about any single individual in the population. We managed to skip this middle step and go straight to collecting only the aggregate information that we wanted.

The randomized polling protocol we have just described is old—it is known as randomized response, and it dates to 1965, decades before the introduction of differential privacy. But it turns out that the protocol satisfies 3-differential privacy—that is, differential privacy in which any bad event you were worried about is at most three times more likely if you participated in the survey.[2] There is nothing special about the number 3 here, though. By decreasing the probability that the protocol asks people to tell the truth, we can make the protocol more private. By increasing the probability, we can make the protocol less private. This lets us quantitatively manage an important trade-off:

[2] In the standard accounting, we would say it satisfies $\ln(3)$ differential privacy, where ln denotes natural logarithm.

the more private we make the protocol, the more plausible deniability each individual we poll gets, and the less anyone can learn from an individual polling response. On the other hand, the smaller the privacy parameter gets, the more error-prone the process of working backward gets—mapping from the proportion of yes answers in our poll to the actual proportion of cheaters in our population. So to get the same accuracy, the smaller the privacy parameter, the more people we need to poll.

Computing an average is the simplest statistical analysis you might want to do—but it is by no means the only one you can do with differential privacy. The biggest strength of differential privacy is that it is compositional, which means it doesn't break the way *k*-anonymity does when you run more than one differentially private analysis. If you've got two differentially private algorithms, you can run them both and the result is still differentially private. You can give the output of one as the input to the other, and the result is still differentially private. This is enormously useful, because it means you can build complicated algorithms by piecing together simple differentially private building blocks. It would be a tedious mess if we needed to reason about the privacy properties of every new algorithm afresh—and it would be a horrendously complex problem if the algorithm was long and complicated.

Instead, we can reason about the privacy guarantees of simple components, such as computing an average. We can then go about designing complicated algorithms by gluing together these simple primitives in various ways, and be guaranteed to still satisfy differential privacy. This makes private algorithm design modular, just like standard nonprivate algorithm design. So we can go from computing averages to performing some optimization over a dataset and then to privately training neural networks. As a general rule of thumb, any statistical analysis or optimization—one whose success is defined

with respect to the underlying distribution, and not with respect to any particular dataset—can be made differentially private, albeit usually with a need for more data.

WHOM DO YOU TRUST?

Randomized response actually satisfies an even stronger constraint than differential privacy. Differential privacy only requires that nobody should be able to guess substantially better than randomly whether a given output was computed with an individual's data or without it. In our polling example, the natural output of the computation is the final average we compute: the estimated proportion of Philadelphia husbands who have cheated on their wives. But randomized response promises not only that the final average is differentially private but also that the entire set of collected records is as well. In other words, randomized response satisfies differential privacy even if we publish the pollster's entire spreadsheet, with all the data she has collected, and not just the final average. It promises differential privacy not just from an outside observer but from the pollster herself.

If we only wanted differential privacy from an outside observer (and trusted the pollster not to leak the raw data), then she could have conducted the poll in the standard way and only added noise just to the final average that she published. The benefit of this approach is that far less noise would be necessary. In randomized response, every person individually adds enough noise to obscure his or her data point. When we average all of the provided data, in aggregate we have added much more noise than we really needed. If we trust the pollster with the real data, she can aggregate the data and then add just enough noise to obscure the contribution of one person. This leads to a more accurate estimate, but of course it doesn't come

for free: it doesn't provide the stronger guarantee of safety if the pollster's records are subpoenaed.

Whether we want this stronger guarantee or not (and whether it is worth the trade-off with error) depends on what our model of the world is and who we think is trying to violate our privacy. The standard guarantee of differential privacy assumes that the algorithm analyzing the data is run by a trusted administrator: that is, we trust the algorithm (and whoever has access to its implementation) to do with our data only what it is supposed to do. The algorithm and the administrator are the ones who add the privacy protections—we trust them to do this. Because we have a trusted administrator who can aggregate all the data "in the clear" (before privacy is added), this is sometimes referred to as the centralized model of differential privacy.

Randomized response instead operates in what is called the local or decentralized model of differential privacy. Here there is no trusted data administrator. Individuals add their own privacy protection locally, in the form of data perturbations, before they hand it over. This is what happens in the randomized response polling method: each person flips his or her own coin and gives a random response to the pollster, and the pollster never sees the true answers. Another suggestive way of thinking about centralized versus local differential privacy is whether the privacy is added on the "server" side (centralized) or the "client" side (local).

In many ways, deciding which trust model you want is a more important decision than setting the quantitative privacy parameter of your algorithm. But the two decisions cannot be made in isolation. Since differential privacy in the local model gives a much stronger guarantee, it is not surprising that choosing it has a substantial cost. In general, for a fixed privacy parameter (say, 2), satisfying differential privacy in the local model will lead to a less accurate analysis than in the centralized trust model. Or, turning this trade-off on its head, for a fixed accuracy requirement, opting for the local model will require either more data than would be needed in the centralized model or else a worse

privacy parameter. These realities play a big role in shaping the three most important deployments of differential privacy to date.

OUT OF THE LAB AND INTO THE WILD

Two of the first large-scale commercial deployments of differential privacy were implemented by Google and Apple. In 2014, Google announced on its security blog that its Chrome browser had started collecting certain usage statistics about malware on users' computers with differential privacy. In 2016, Apple announced that iPhones would begin collecting usage statistics using differential privacy. These two large-scale deployments have many things in common. First, both Google and Apple decided to operate in the local trust model, basing their algorithms on randomized-response-like methods. This means that Google and Apple are never collecting the relevant private data itself. Instead, your iPhone is simulating coin flips to run a randomized response protocol and sending to Apple only the random outputs generated. Second, both deployments were used to privately gather data that the companies had previously not been gathering at all, rather than implementing differential privacy on top of datasets that they already had available to them.

For Google and Apple, the trade-offs that go with the local model of privacy make sense. First, neither company is necessarily trusted by its users. No matter what you think about the companies themselves, there is a real risk that data they store on their servers will be revealed to others via hacking or government subpoena. For example, in 2013, Edward Snowden released thousands of documents revealing intelligence-gathering techniques used by the National Security Agency (NSA). Among the revelations were that the NSA had been eavesdropping on communications that flowed between Google (and Yahoo) data centers, without Google's knowledge. Security has been tightened since then—but whether or not surreptitious data exfiltration has been stopped, there is still much data released to national governments

via the ordinary legal process. Google reports that in the one-year period starting July 2016, government authorities requested data for more than 157,000 user accounts. Google produced data in response to roughly 65 percent of these requests. A guarantee of differential privacy in the centralized model promises nothing against these kinds of threats: so long as Google or Apple holds customer data, it is subject to release through nonprivate channels.

On the other hand, in the local model of differential privacy, companies such as Google and Apple never have to collect the private data in the first place. Instead, they play the role of the pollster in our example above, only recording the noisy responses of users who always maintain a strong guarantee of plausible deniability. The pollster might lose her spreadsheet to a hacker who breaks into her computer—but the spreadsheet doesn't contain much information about any particular person. A divorce lawyer might subpoena the record of how his client's husband answered the pollster's question—but the answer he gets back doesn't tell him what he wanted to know. Of course, we've seen that asking for this stronger subpoena-proof form of privacy protection comes at a cost: to get an accurate picture of population-wide statistics, you need a *lot* of data. But both Google and Apple are in a strong position to make this trade-off. Google has more than a billion active Chrome users, and Apple has sold more than a billion iPhones.

It is also noteworthy that both Google and Apple applied differential privacy to data that they weren't collecting before, rather than to data sources they already had available. Adding privacy protections can be a tough sell to engineers who already have the data available to them. Asking for privacy almost always means a real, measurable degradation in data quality in exchange for a decrease in risk that can feel much more nebulous. It is hard to convince an engineering team to give up access to a clean data source they already have. But if differential privacy is framed as a way to get access to new data sources—data that previously was not being collected at all, because of privacy

concerns—that is a different story entirely. When framed this way, differential privacy is a way to get more data, not an obligation that degrades existing analyses. This is what happened at both Google and Apple.

A third large-scale deployment of differential privacy serves as an interesting contrast to the Google and Apple use cases. In September 2017, the US Census Bureau announced that all statistical analyses published as part of the 2020 Census will be protected with differential privacy. In contrast to the large industrial deployments, the Census Bureau will operate using the centralized model of differential privacy, collecting the data (as it always has) exactly and adding privacy protections to the aggregate statistics it publicly releases. Moreover, this isn't letting the Census Bureau access a new data source—the requirement to conduct a census every ten years is laid out in the US Constitution, and the first census was conducted in 1790.

So why did the Census Bureau decide to adopt differential privacy in the first place? And why did it opt for the higher accuracy that comes with the weaker centralized trust model, which does not protect against subpoenas, hacks, and the like?

The answer to the first question is that the Census Bureau is legally mandated to take measures to protect individual privacy. For example, all Census Bureau employees take an oath of nondisclosure and are sworn for life to protect all information that could identify individuals. There is no option to release information in the clear: privacy is a must, and the only question is what privacy protections should be used. So the alternative wasn't to do nothing about privacy—it was to continue taking the heuristic approaches of prior censuses that had no hard promises about privacy and difficult-to-quantify effects on data accuracy. There is no technology other than differential privacy that gives similarly principled, formal privacy guarantees while preserving the ability to estimate population-wide statistics.

The answer to the second question—about adopting the centralized model—is that legal protections give the Census Bureau a much

stronger claim to being a trusted administrator than Google or Apple. By law, the Census Bureau cannot share individual data records—that is, your responses to questions on the 2020 Census—even with any other government agency, including the IRS, the FBI, or the CIA.

WHAT DIFFERENTIAL PRIVACY DOES NOT PROMISE

At its core, differential privacy is meant to protect the secrets held in individual data records while allowing the computation of aggregate statistics. But some secrets are embedded in the records of *many* people. Differential privacy does not protect these.

The fitness tracking website Strava is a dramatic example of this. It allows users to upload data from devices such as Fitbits to their website, so that they can keep track of their activity history and locations. In November 2017, Strava released a data visualization of the aggregate activity of its users, consisting of more than three trillion uploaded GPS coordinates. This lets you see, for example, popular running routes in almost every major city. But large parts of the globe are almost devoid of activity on this map—for example, poor and war-torn regions including Syria, Somalia, and Afghanistan. Most people living in these places don't have Fitbits or use Strava.

But there is a notable exception in each of these countries: US military personnel. The US military encourages its soldiers to use Fitbits, and they do. The Strava data reveals (in aggregate) the most popular jogging routes in each of these locations, as it is intended to do. But the most popular jogging routes in Afghanistan's Helmand Province turn out to be on military bases, not all of which are supposed to be publicly known. The Strava data revealed sensitive state secrets, even though it did not necessarily reveal any individual secret. Data such as the Strava location heatmap could have been assembled with differential privacy protections, but this would only promise that the map would be nearly unchanged if any single soldier had decided not

to use his Fitbit—it does *not* promise that the aggregate behavior of soldiers on a military base would be hidden. (It's worth noting that differential privacy does allow the data of small groups to be protected—in particular, an algorithm promising 3-differential privacy for individuals also provides 3^k-differential privacy for the data of sets of k individuals—but if k is large, as with an entire platoon of soldiers, this won't be a very meaningful guarantee.)

Differential privacy can also allow people to learn about secrets that you think of as your own, if it turns out that they can be deduced from publicly available data—often in ways that you could not anticipate. Remember that by design, differential privacy protects you from the consequences of what an observer might be able to learn specifically because of your data but does not protect you from what an attacker might be able to infer about you using general knowledge of the world. So it prevented an observer from deducing that Roger had lung cancer as a result of his participation in the British Doctors Survey, but it didn't prevent us from learning that smoking was a good predictor of lung cancer—even though this might cause us to change our beliefs about Roger's risk for lung cancer if we know he is a smoker. In general this is a good thing—it lets us still learn about the world and conduct science while being able to meaningfully protect privacy.

But the world is full of curious correlations, and as machine learning gets more powerful and data sources become more diverse, we are able to learn more and more facts about the world that let us infer information about individuals that they might have wanted to keep private. In the simple example of the British Doctors Survey, if only Roger knew the conclusion of the study ahead of time, he could have tried to keep his smoking habits private. But how was he to know? Similarly, we make all sorts of seemingly innocuous information about ourselves public, but this opens us up to a surprisingly large number of inferences.

For example, Facebook offers many options allowing us to either hide or reveal information about ourselves on our Facebook profiles.

If we want, we can decide not to show our sexual orientation, our relationship status, or our age on our public profiles. This gives users a sense of control. But by default, Facebook makes public what pages we "like." For example, you might like pages for *The Colbert Report*, curly fries, and Hello Kitty. This seemingly trivial stream of data gives your user profile on Facebook a statistical fingerprint that can be correlated with other things. In 2013, a team of researchers at Cambridge University showed that this stream of data could be mined to uncover correlations that would accurately map people's seemingly innocuous likes to properties about them that they may have wanted to keep private. From a user's likes, the researchers were able to guess at the user's gender; political affiliation (Republican or Democratic); sexual orientation; religion; relationship status; use of drugs, cigarettes, and alcohol; and even whether the user's parents divorced before he or she turned twenty-one—all with a statistically significant level of accuracy. These are all things that you might have wanted to keep private, or at least not post publicly on the Internet for everyone to see. But you probably didn't worry about hiding that you like curly fries.

In a similar demonstration in 2010, a machine learning start-up called Hunch released a demo that it called the Twitter Predictor Game. It was simple: you told it your Twitter handle, and it scoured the Twitter network to gather some simple data, consisting of whom you followed and who followed you. From this apparently innocuous data, it then asked you a series of private questions—things you likely never tweeted about (it didn't read your tweets anyway)—and it would guess the answers. It could accurately guess whether you were pro-choice or pro-life. Whether you thought gay marriage should be legal. Whether you had a brokerage account. Whether you listened to talk radio. How often you wore fragrance. Whether you had ever purchased something from an infomercial. Whether you preferred Star Wars or Star Trek. Whether you had ever kept a journal.

Fig. 4. Hunch would ask users a wide variety of questions and then predict their answers from their pattern of followers on Twitter.

It was shockingly accurate—it correctly predicted user answers about 85 percent of the time. It wasn't exploiting your private data to answer these questions, because you never gave it access to data explicitly answering those questions. Data explicitly answering those questions didn't exist. Instead, it was looking only at data you had made public—the list of people you followed on Twitter—and making inferences from information it had about the population at large. It was doing social science on a large scale—it was learning about the world—and by design, differential privacy is not something that would protect you from people making these kinds of inferences about you.

These two examples are nothing more than the modern version of knowing that smoking increases one's risk of lung cancer and then making inferences about the cancer risk of a known smoker. But as machine learning becomes more powerful, we are able to discover increasingly complicated and non-obvious correlations. Seemingly useless bits of data—the digital exhaust that everyone is constantly producing as they use the Internet, purchasing items, clicking on links, and liking posts—turn out to be surprisingly accurate predictors

of other things that we might really wish that other people didn't know. Differential privacy doesn't protect you from these kinds of inferences, because the correlations they rely on would exist whether or not your data was available. They are just facts about the world that large-scale machine learning lets us discover. And this is by design—these kinds of privacy risks will always exist whenever people are allowed to study the world and draw conclusions. There isn't any way to stop it, short of hiding all data and stopping science altogether.

Let's end the chapter with one more example that demonstrates how facts about the world and facts about you can sometimes be entangled. Your DNA represents some of the most sensitive information that there is about you. In a very real sense it encodes your identity: what you look like, where your ancestors came from, and what diseases you might have or be susceptible to. And it can also be used as forensic evidence to place you at the scene of a crime, and ultimately to convict and imprison you. But your DNA is not yours alone: you share much of it with your parents, your children, your siblings, and, to a lesser extent, all of your relatives up and down your family tree. So although we think of your DNA as *your* data—something you are free to make public or keep secret at your discretion—what you do with it affects other people as well. For example, if you have committed a crime and are on the run from police, you would be foolish to upload your DNA to a publicly searchable database, because the police could match the public record to a sample from the crime scene and apprehend you. But you might not be able to stop your relatives from uploading *their* genetic information.

This is exactly how the infamous Golden State Killer was captured in 2018. He had been linked to more than fifty rapes and twelve murders between 1976 and 1986, leaving plenty of DNA evidence behind—but since he wasn't in any law enforcement database, the trail had gone cold and police had been unable to find him. But times have changed, and law enforcement DNA databases are no longer the only game in

town. Starting in the mid-2000s, people began to voluntarily upload their own DNA to public databases on the Internet, to learn more about their family histories. For example, in 2011, two volunteers started a website called GEDmatch, which hobbyists could use to upload DNA profiles that they had generated using commercial sites such as 23andMe. Users could search for partial matches, ostensibly to find their own distant relatives and link their family trees, which GEDmatch made available. But anyone else could also conduct such a search—and police still investigating the Golden State Killer uploaded a sample of his DNA, taken from a crime scene, in hopes of finding a match. And they did—not to the Golden State Killer himself, who hadn't uploaded his own DNA, but to a number of his relatives, who had. From their family trees, the police were able to find a small list of suspects, ultimately arresting Joseph James DeAngelo at age seventy-two.

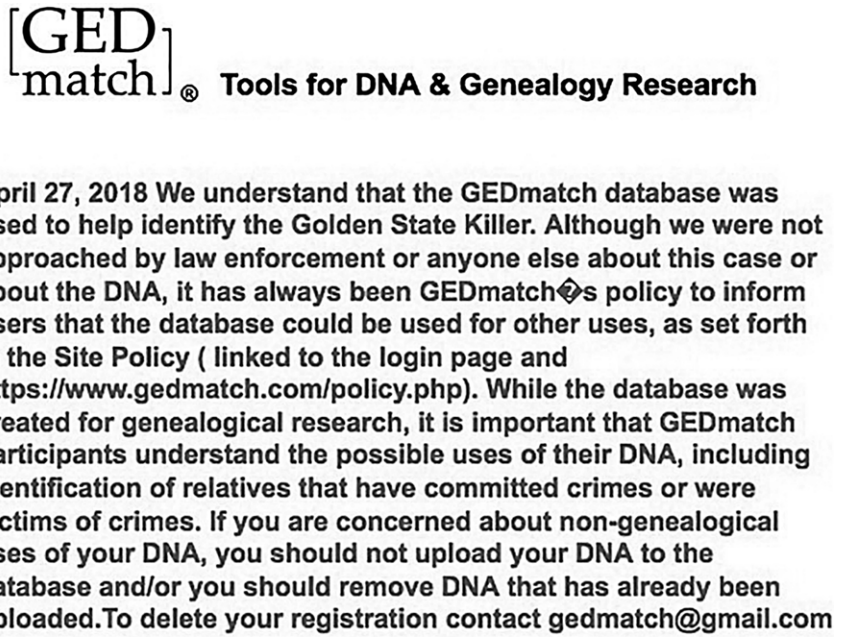

Fig. 5. A message posted on the GEDmatch website to its users after the Golden State Killer was arrested thanks to DNA data found on its website.

The case of the Golden State Killer raises a number of difficult privacy challenges, because genetic data brings into sharp relief two related things we think of as personal rights: the autonomy to do what we like with "our own" data, and the ability to control our "private" data. Because your DNA contains information not just about you but about your relatives, these two things are in conflict: your "private" data is not always "your own." As Erin Murphy, a law professor at New York University, told the *New York Times*, "If your sibling or parent or child engaged in this activity online, they are compromising your family for generations." And because of this, this is not the kind of privacy threat that differential privacy is designed to protect against. Differential privacy controls the difference between what can be learned about you when you make the individual decision to allow your data to be used and what can be learned when you don't. So it gives you little incentive to withhold your data. But this might be because, to take the case of a genetic database, much about you would become known even if you did withhold your data, because some of your relatives freely provided theirs. This kind of reasoning about how the effect of your own actions in isolation affects incentives is the domain of game theory, and the topic of Chapter 3. Because differential privacy is based on the same basic idea, it might not come as a surprise that tools from differential privacy will be useful in game theory as well.