Privacy Enhancing Technologies FS2025 Lecture 20-21-22 – Data Anonymization and Reconstruction Attacks

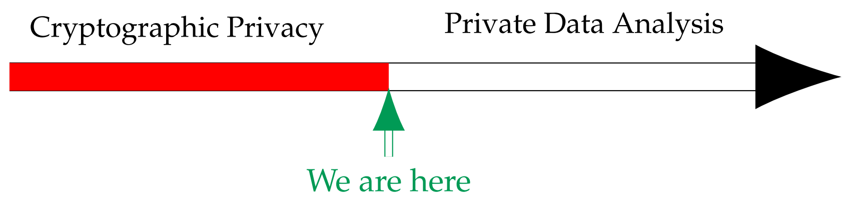
Florian Tramèr

These lecture notes are partially inspired by those of Gautam Kamath.[1]

AGENDA

1. Recap on cryptographic privacy
2. Private data analysis
3. The limits of data anonymization
4. Interactive data reconstruction attacks
5. Reconstruction attacks in practice

1 Where Are We?



We've roughly reached the midpoint of this class. So far, we have focused on cryptographic approaches to privacy. Informally, these approaches all aim to compute some function f over multiple parties' data, while revealing *no more information than the output of f*.

But what if this isn't enough? What if the output of the function itself reveals sensitive information about the data of one or more parties? In particular, what happens if we start computing *many* functions f over parties' data?

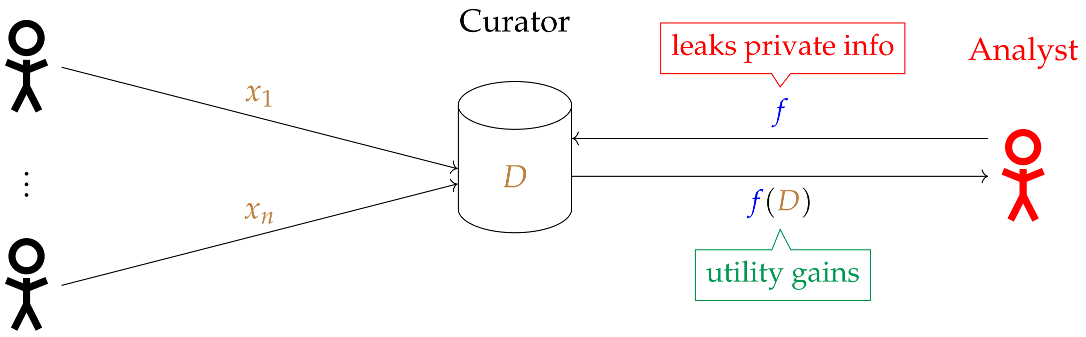
Here we enter the realm of *private data analysis*, which will be our focus for the rest of this course. From now on, we will primarily focus on *statistical* (a.k.a. information-theoretic) notions of privacy. We will still consider adversaries, but these adversaries will typically not be computationally bounded.

2 The Private Data Analysis Setting

The general setting we consider is that some users provide their data to a data curator. The curator then makes some of this data available to a data analyst who aims to learn interesting things about the data.

[1] http://www.gautamkamath.com/CS860notes/lec2.pdf

We will typically assume that the curator is a trusted party, and that the analyst is not. There is thus a tension between the utility of the data and its privacy: giving the analyst more information about the data will make it easier for them to learn interesting things that could benefit the users, but it will also leak more information about the users' data.



Note that cryptographic approaches such as MPC won't help us in providing privacy *with respect to the analyst*. Indeed, the analyst already interacts with a trusted party (the curator). The privacy issue is precisely that they perform some analysis on the data.

Of course, cryptography could help us get rid of the trust assumption in the curator. We'll get back to this in a future lecture.

3 Non-interactive Data Analysis: Anonymized Data Release

Let's start with the simplest setting, where the curator just wants to release data once-and-for-all to the analyst(s). This does not seem great for privacy, but it is a very common setting in practice. Indeed, setting up some kind of data analysis interface that meaningfully limits the type of queries that the analyst can ask is a lot of work.

To "preserve" privacy in such a setting, a common approach is to *anonymize* the data. That is, the curator releases (parts of) the data in the clear, but aims to remove any information that could be used to link the data to a specific individual. So, for example, a hospital might release a dataset of patient diagnoses, but redact or approximate names, ages, and other identifying information. This is a nice desiderata, but unfortunately history is riddled with examples showing that this is essentially impossible to achieve.

3.1 Famous Anonymization Failures

The NYC Taxi Data. In 2014, the New York City Taxi and Limousine Commission released a dataset of taxi rides, including the pick-up and drop-off locations, the date and time of the ride, and the fare. To attempt to anonymize the data, the driver's medallion and license numbers were replaced by pseudo-random numbers like `6B111958A39B24140C973B262EA9FEA5`.

While inspecting the data, it was found that the medallion `CFCD208495D565EF66E7DFF9F98764DA` has an unusually high number of rides. Later, someone found the reason: this is the MD5 hash of "0", which was presumably used to represent a missing value. So the anonymization method was just hashing the values with MD5. *Hashing values is not a good way to hide them*, recall the lecture on commitment schemes. Since the medallion and license numbers are quite short, it was easy to just enumerate all their hashes and recover the original values.

A (slightly) better approach would have been to replace each number with a truly random identifier. But this could still lead to some privacy issues. E.g., if you see someone enter a taxi at a specific location and time, you can then go check the database to find their ride and see where they went. This is an example of a privacy attack that exploits *side-information*.

The Netflix Prize. In 2006, Netflix launched a competition with a prize of $1 million USD to improve their movie recommendation system. They released a dataset of movie ratings with a pseudonymous user ID (i.e., if a user rated multiple movies, the same random ID would be used for each rating).

In a landmark paper, Narayanan and Shmatikov [NS08] showed that it was possible to de-anonymize parts of this dataset by cross-referencing it with publicly available data. Specifically, they used movie ratings from IMDb (which are not anonymized) to find a Netflix user with a highly similar rating profile, and then infer other movies this user has seen (but not disclosed on their public IMDb profile).

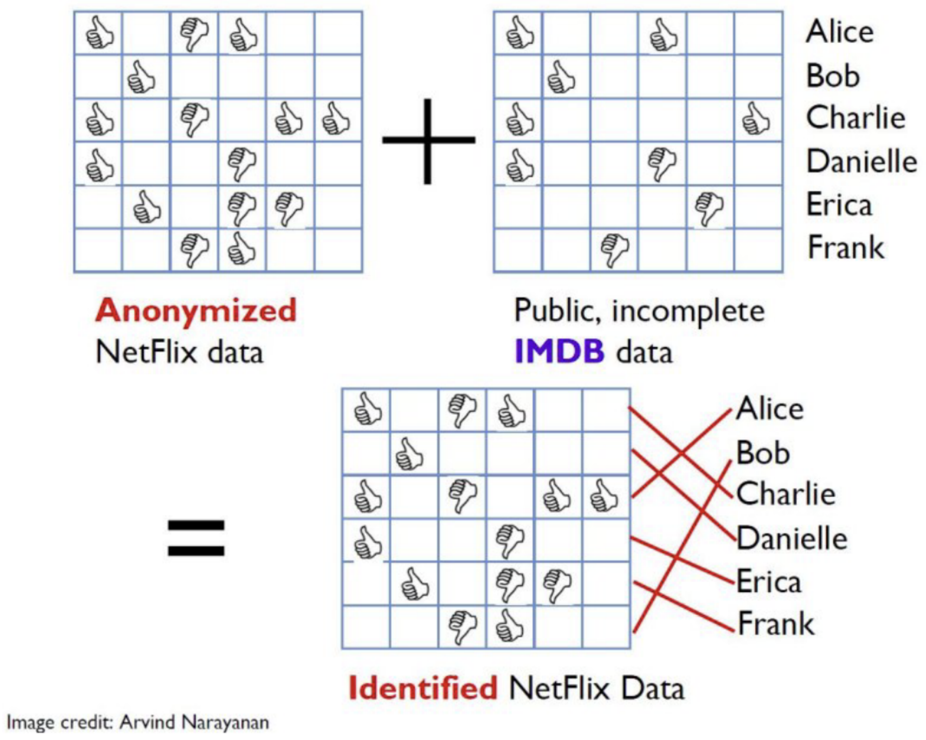


Figure 1: An illustration of the attack by Narayanan and Shmatikov [NS08].

The Massachusetts Group Insurance Commission. In the 1990s the Massachusetts Group Insurance Commission started giving researchers access to (anonymized) hospital visit records for all state employees. The Massachusetts governor, William Weld, promised that patient privacy would be protected. And obviously, politicians always tell the truth.

The data was anonymized by removing identifying features such as name and social security number, while retaining features such as date of birth, gender, ZIP code, and diagnosis.

Latanya Sweeney, a PhD student at MIT, wanted to prove that the anonymization was not effective. She first calculated that there are roughly $365 \cdot 78 \cdot 2 = 56{,}940$ unique combinations of birth date and gender for people up to 78 years old. Yet, the average ZIP code in the US only has about 25,000 people. So these three features might be enough to uniquely identify most people (87% of the US population to be precise [Swe00]).

To test this, she checked William Weld's birth date and ZIP code (which were easily available in the public record), and found a single match in the anonymized hospital data [Swe15]. Her results and analysis had a wide impact on privacy regulations and policy—and were rejected for academic publication more than 20 times...

3.2 K-Anonymity

As an attempt to address these issues, Sweeney proposed the concept of *k-anonymity*, a property of a dataset that ensures that each individual's data is indistinguishable from at least $k - 1$ other individuals' data.

Definition 1 (k-Anonymity [Swe02])**.** A dataset is *k-anonymous* if for each individual, there are at least $k - 1$ other individuals in the dataset that have the same value for all "quasi-identifiers."

The definition refers to "quasi-identifiers", which are features that can be used to partially identify an individual (e.g., ZIP code, date of birth, gender). These are in contrast to "identifiers" that are unique to an individual (e.g., name, social security number) and "non-identifiers" which are assumed to be unknown to an adversary or unrelated to the user's identity (e.g., disease diagnosis, pseudonym, etc.).

Example. Consider the dataset below. This provides 1-anonymity (i.e., no anonymity at all) since each individual is identifiable by their unique name.

Identifier	Quasi-identifiers				Non-iden.
Name	**Age**	**Gender**	**Weight**	**ZIP-code**	**Disease**
Alice	25	F	65	12345	Flu
Bob	30	M	75	12671	Heart
Charlie	20	M	90	23456	Cancer
David	47	M	92	20451	Cancer
Eve	52	F	49	12800	Virus
Frank	16	M	75	12345	Flu
Grace	61	F	53	24889	Virus
Haley	31	F	71	80191	Cancer
Irene	38	F	62	81976	Heart
Jennifer	63	F	55	22288	Cancer

The table below provides 2-anonymity with respect to the identifiers and quasi-identifiers. To achieve this, we have combined *suppression* (i.e., replacing a value with a "*"), and *generalization* (i.e., replacing a value with a range of values).

Name	Age	Gender	Weight	ZIP-code	Disease
*	25–40	F	60–70	*****	Flu
*	15–30	M	75	12***	Heart
*	20–50	M	80–95	2****	Cancer
*	20–50	M	80–95	2****	Cancer
*	30–55	F	45–75	*****	Virus
*	15–30	M	75	12***	Flu
*	60–65	F	50–80	2****	Virus
*	30–55	F	45–75	*****	Cancer
*	25–40	F	60–70	*****	Heart
*	60–65	F	50–80	2****	Cancer

Attacks. Unfortunately, k-anonymity still allows for a wide range of attacks:

- *Homogeneity attack.* If all individuals in an anonymity group have the same sensitive attribute, then this attribute can be inferred. E.g., in the table above, suppose you know that Charlie who weighs 90kg is in the dataset. Then you can infer that he has Cancer.
- *Background knowledge attacks.* E.g., suppose you know that Bob who lives in the ZIP code 12671 is in the dataset. You also happen to know that he has been vaccinated against the flu. Then you can be fairly certain that he has a heart condition.
- *Composition attacks.* If you have two k-anonymous datasets, the composition of these datasets might provide no anonymity at all.

  E.g., suppose the hospital releases a second 2-anonymous dataset about the same patients, but this time generalizing different attributes:

Age	Gender	ZIP-code	Disease
20–55	F	12***	Flu
15–35	M	12***	Heart
15–25	M	2****	Cancer
15–25	M	2****	Flu
20–55	F	12***	Virus
15–35	M	12***	Flu
60–70	F	2****	Virus
30–40	F	8****	Cancer
30–40	F	8****	Heart
60–70	F	2****	Cancer

  Suppose you know that Alice is a 25-year-old woman. Based on the first table alone, her assigned disease is either Flu or Heart. Based on the second table alone, it is either Flu or Virus. By intersecting these two possibilities, we find that the only disease that appears in both groups is Flu—revealing Alice's diagnosis.
- *Downcoding attacks.* By using properties of the algorithm that produces the anonymized data, an adversary might infer something about the original data [Coh22]. For example,

anonymization algorithms don't generalize values more than necessary, so the generalized value leaks information about the original values.

4 Attacks on Aggregate Statistics

So clearly releasing a bunch of data, even anonymized, is not a good idea. Instead, we'll now consider a setting where we only provide *aggregate* statistics over all the data. This is a common setting in practice, e.g., in medical studies, surveys, census data, etc.

Let's consider a census (e.g., as conducted in the US every ten years) which collects individual information such as age, gender, ethnicity, marital status, etc. As this data may be too sensitive to release directly (as we just saw), the census aggregates this information across different subsets of a population, and releases only those aggregates.

As a (highly) simplified example, consider the census information below collected from a small "block" of 7 people. We aggregate their average and median age, further broken down by gender, ethnicity, and marital status. Any statistic that is computed over only 1 or 2 people is suppressed (marked as (D)) to provide a form of k-anonymity.

			Age	
Statistic	**Group**	**Count**	**Median**	**Mean**
1A	Total Population	7	30	38.0
2A	Female	4	30	33.5
2B	Male	3	30	44.0
2C	Black or African American	4	51	48.5
2D	White	3	24	24.0
3A	Single Adults	(-)	(-)	(-)
3B	Married Adults	4	51	54.0
4A	Black or African American Female	3	36	36.7
4B	Black or African American Male	(-)	(-)	(-)
4C	White Male	(-)	(-)	(-)
4D	White Female	(-)	(-)	(-)
5A	Persons Under 5 Years	(-)	(-)	(-)
5B	Persons Under 18 Years	(-)	(-)	(-)
5C	Persons 64 Years or Over	(-)	(-)	(-)

Table 1: Fictitious statistics of a Census block, from [GAM19]. (-) indicates that the statistic has been suppressed because it is an aggregate from only 1 or 2 people.

Attacks. The different statistics computed over the same population give us a lot of redundant information. It turns out that we can use this redundancy to recover *all* information about the population!

Let's start with an example:

> From statistic 2B, we know there are 3 males. Denote their ages as $A \leq B \leq C$.
>
> Assuming ages are integers between 0 and 125, this would mean there could be as many as $126^3 \approx 2$ million possible combinations of A, B, C.
>
> But we know that their average age is 44 and the median age is 30. This immediately tells us that $B = 30$, $A \leq 30$, and $C \geq 30$.
>
> We then have $(A + B + C)/3 = 44$, or equivalently $A + C = 102$. This reduces the number of combinations to just 31.

So by looking at a single constraint, we recovered the exact age of one person and reduced the number of possible combinations for the other two by 99.99%.

We can automate this process by writing out constraints for all statistics computed over the same population, and then solving the constraint system to find a solution. Depending on the type of constraints we use, this problem is NP-Hard. But real-world data is rarely worst-case and there are a number of heuristic solvers that can tackle such problems with millions of constraints. For the statistics in the census example above, it turns out that a single solution exists, which can be found on a laptop in 0.1 seconds [GAM19].

5 How Many Queries Can You Answer Privately?

So what went wrong with the above data release? Essentially, it's a matter of releasing *too many*, *highly accurate* statistics about the population.

We will now work towards a formal model of data analysis that will allow us to prove bounds on what can and cannot be computed privately. This model will be rather simplistic, but it will allow us to prove some interesting impossibility results.

We assume the curator's database consists of n records (or "rows") of multiple attributes (the "columns"). We also assume that all attributes except one are public (i.e., they are known to the adversary), and the private attribute is a single bit. We will look at attacks where the adversary knows all the public attributes (also called identifiers), and aims to reconstruct the private bits.

	$\overbrace{\qquad\qquad}^{\textit{Public attributes}}$		$\overbrace{\qquad}^{\textit{Private attribute}}$
Name	**Age**	**Gender**	**Disease?**
Alice	25	F	1
Bob	30	M	0
...	...	...	...
Zack	18	M	1

$\Longrightarrow$

Identifiers	**Secret**
z_1	s_1
z_2	s_2
...	...
z_n	s_n

The adversary gets to ask a series of *queries* over this database, which the curator answers (possibly with noise). Specifically, our adversary will be allowed to ask *subset queries*, which sum up the private attribute for a subset of the rows. E.g., *"How many women under 25 in the database have the disease?"*

Or in (pseudo)-SQL form:

```
SELECT SUM(Disease) FROM database WHERE Gender = F AND Age < 25;
```

We will abstract this away as follows:

- Each individual's data in the database $D = (z, s)$ is a pair (z_i, s_i), where $z_i \in \mathcal{Z}$ is the identifier and $s_i \in \{0, 1\}$ is the private bit. E.g., in our example above, we have $\mathcal{Z} = \{\text{names}\} \times \{\text{ages}\} \times \{\text{genders}\}$. We assume that all the identifiers are distinct, i.e., $z_i \neq z_j$ for $i \neq j$, and known to the adversary.
- The adversary makes *counting queries*, which are functions of the form

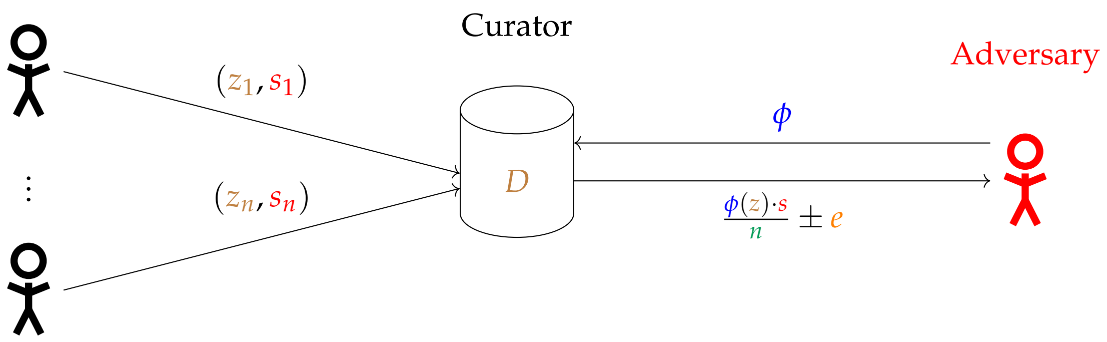
$$\begin{aligned} f_\phi(D) &= \frac{1}{n} \sum_{i=1}^{n} \phi(z_i) \cdot s_i \\ &= \frac{\phi(z) \cdot s}{n} \quad \text{for some predicate } \phi : \mathcal{Z} \to \{0, 1\}. \end{aligned}$$

  which return the normalized[2] sum of the private attribute for the subset of rows that satisfy the predicate ϕ. This can be written as the dot product of the predicate vector applied to each row $\phi(z)$ and the private attribute vector s.
- The curator answers the query by returning $\texttt{Ans}(\phi) = f_\phi(D) \pm e$, where e is some noise.[3] We assume that the noise is bounded, i.e., $0 \leq e \leq E$, but is otherwise arbitrary.

[2]This normalization just scales all results by $1/n$ and is not really important for the analysis. But it will make things easier when we look at some results from differential privacy.

[3]If there were no noise, the adversary could easily reconstruct the i-th bit of D by asking a query where $\phi(z) = 1$ if and only if $z = z_i$.

The picture below illustrates this setup.



So let's see what's possible in this setting. That is, how many queries can we answer, and with what noise, before we start leaking "too much" information to the adversary?

How do we define what it means for the adversary to learn "too much" information? For now, we'll content ourselves with a very weak goal: we want to make sure that the adversary cannot *perfectly reconstruct a large chunk of the database*.

Definition 2 (Blatant *non*-privacy. [DN03]). The curator's algorithm `Noisy` is *blatantly non-private* if the adversary can, with *high probability* $1 - o(1)$,[a] construct a database $\tilde{s} \in \{0,1\}^n$ such that s and $\tilde{s}$ differ in $o(n)$ coordinates.

[a]That is, the probability goes to 1 as the database size n goes to infinity.

A blatantly non-private algorithm is pretty disastrous for privacy (hence the name). It means that the adversary can reconstruct more than any constant fraction (e.g., 99.99%) of the database exactly. We definitely don't want that...

A celebrated set of results of Dinur and Nissim [DN03] shows that if we want to answer many queries and avoid blatant non-privacy, then we necessarily need to add a large amount of noise. These results then paved the way to the formalization of differential privacy, which we will discuss in the next few lectures.

2^n subset queries cannot be answered privately with $o(1)$ noise. The first result of Dinur and Nissim considers an adversary who can ask a very large number of queries, *exponential* in the size of the database (actually every single counting query!).

Theorem 1 (Dinur and Nissim, 2003). If the adversary can ask 2^n subset queries on a database of size n, and the curator adds noise of size at most E to each query, then the adversary can output a guess $\tilde{s} \in \{0,1\}^n$ such that s and $\tilde{s}$ differ in at most $4En$ positions.

As an example, suppose we set $E = 1/401$. This is a rather large amount of noise: for any query on a subset of less than $1/400$-th of the data, the noise will drown out any signal. And yet, even with this large amount of noise, an adversary can reconstruct the database on 99% of the entries. And if we set a smaller noise, $E = o(1)$, then the curator's algorithm is blatantly non-private!

Proof. The adversary's algorithm is very simple: for each vector $v \in \{0,1\}^n$, make a counting query with $\phi_v(z_i) = v_i$ and obtain the noisy answer $\texttt{Ans}(\phi_v) = \frac{\phi_v(z)\cdot s}{n} \pm e$. That is, we simply count the secrets over all possible subsets of the users.

Then, output a guess $\tilde{s}$ that is *consistent* with all the queries, i.e.,

$$\left| \frac{\phi_v(z) \cdot \tilde{s}}{n} - \texttt{Ans}(\phi_v) \right| \le E \quad \forall v \in \{0,1\}^n .$$

First, it's easy to see that this algorithm always terminates, since the true secret vector s is consistent with all queries.

Now, let's show that the guess output by the adversary differs from the true secrets s in at most $4En$ positions, i.e.,

$$\texttt{dist}(s, \tilde{s}) := \sum_{i=1}^{n} \mathbb{1}[s_i \ne \tilde{s}_i] \le 4En.$$

We proceed by contradiction. Suppose that $\texttt{dist}(s, \tilde{s}) > 4En$. We now split the errors into two parts:

$$S_0 = \{i \mid s_i = 0, \tilde{s}_i = 1\} \quad \text{and} \quad S_1 = \{i \mid s_i = 1, \tilde{s}_i = 0\}.$$

Since we have that $|S_0| + |S_1| = \texttt{dist}(s, \tilde{s}) > 4En$, one of the two sets must be larger than $2En$. Without loss of generality, let's assume that $|S_0| > 2En$.

Denote by v the *indicator vector* of S_0, i.e., $v_i = 1$ if $i \in S_0$ and $v_i = 0$ otherwise. Note that $\phi_v(z) \cdot s = 0$ since S_0 consists only of positions where $s_i = 0$. And so it must be that the noisy answer to ϕ_v satisfies $\texttt{Ans}(\phi_v) \le E$.

On the other hand, $\frac{\phi_v(z) \cdot \tilde{s}}{n} = \frac{|S_0|}{n} > 2E$. And so we have

$$\left| \frac{\phi_v(z) \cdot \tilde{s}}{n} - \texttt{Ans}(\phi_v) \right| > E \, ,$$

which contradicts the fact that $\tilde{s}$ is consistent with all queries. □

n subset queries cannot be answered privately with $o(1/\sqrt{n})$ noise. To recap, the previous result shows that if the adversary can ask a counting query over all 2^n subsets of the database, then we need noise of magnitude $\Omega(1)$ to prevent the adversary from reconstructing essentially all the secrets. This is pretty bad: the only way to prevent a flagrant breach of privacy is to add noise that essentially drowns out any signal (the true answer is always in the range $[0, 1]$). But this requires the adversary to make an exponentially large number of queries.

The next result shows that if the adversary only asks a *linear* number of counting queries, then noise of magnitude $\Omega(1/\sqrt{n})$ is necessary for privacy.

Theorem 2 (Dinur and Nissim, 2003)**.** If the adversary can ask n counting queries on a database of size n, and the curator adds noise of size bounded by $E = o(1/\sqrt{n})$ to each query, then the curator's algorithm is blatantly non-private.

Proof (sketch). The adversary's algorithm is similar as before, except that the adversary now makes n *random* counting queries (i.e., on predicates that select each individual user independently with probability 1/2). The adversary then outputs a vector $\tilde{s}$ that is consistent with all noisy responses. This can be done by solving a *linear program* over the unknown variables s_i (the private bits of the database). The attack thus runs in polynomial time.

We will switch here to a rescaled setting where the secrets s, predicate vector $\phi(z)$ and guess $\tilde{s}$ are vectors in $\{-1, 1\}^n$ (where $\phi(z_i) = 1$ if the i-th individual satisfies the predicate ϕ). The true answer is $\texttt{Ans}(\phi) = \frac{1}{n}\phi(z) \cdot s$, and the curator returns a noisy version $\texttt{Noisy}(\phi)$ of this answer. The idea is that if $\tilde{s}$ is far from the true secrets s, then for a random query ϕ, the answers $\frac{1}{n}\phi(z) \cdot s$ and $\frac{1}{n}\phi(z) \cdot \tilde{s}$ will be sufficiently far that they cannot be reconciled with noise of magnitude $o(1/\sqrt{n})$.

Consider the value $\Delta = (\tilde{s} - s) \cdot \phi(z)$, where $\phi(z)$ is a uniformly random predicate. Suppose that $\tilde{s}$ and s differ in $\Omega(n)$ positions. Then the value Δ is the sum of $\Omega(n)$ independent values ± 2, which is a rescaled and shifted version of a $\texttt{Binomial}(\Omega(n), 1/2)$. Notably, Δ has mean 0 and standard deviation $\Omega(\sqrt{n})$, and by *anti-concentration* of the Binomial distribution, it will take a value of magnitude $\Omega(\sqrt{n})$ with high probability. Thus the difference between $\frac{1}{n}\phi(z) \cdot s$ and $\frac{1}{n}\phi(z) \cdot \tilde{s}$ is at least $\Omega(1/\sqrt{n})$.

So $\tilde{s}$ cannot be consistent with $\texttt{Noisy}(\phi)$ if the amount of noise is $o(1/\sqrt{n})$. With a bit more work (see [DN03]), we can show that *every* guess that is far from s will be ruled out with high probability, and so the guess output by the adversary must be close to s. □

6 Linear Reconstruction Attacks in Practice

These attacks are quite devastating, but the setup might seem somewhat unrealistic. The sensitive information of each user is a single bit, and the attacker can make arbitrary subset queries over the database (this somehow implies the attacker knows all the public contents of the database).

It turns out these are just artifacts of the relatively simple formal model we were using, which enables relatively clean proofs. Variants of these attacks do work in practice, and have been used to reconstruct real databases. Let's look at some examples:

The Aircloak challenge. In 2017, Aircloak—a German data anonymization company—ran a bug bounty challenge to reconstruct parts of a database by interacting with their "privacy preserving" query system *Diffix*. This system used a bunch of heuristics to add noise to queries, with the amount of noise increasing with how "specific" the query was (i.e., how many different attributes were used in the query).

In a nice attack, Cohen and Nissim [CN18] showed that an analyst could craft queries that only used a small number of attributes (and thus resulted in low noise), and that would sum up the values of a (roughly) random subset of the database. This then allowed them to apply the attack from Theorem 2 to reconstruct nearly the entire database (and win $5′000)!

The US Census. In Section 4 we looked at a fictitious example of a reconstruction attack for census data. However, researchers at the US Census have actually simulated such attacks on full census data, and found that their attacks could reconstruct the individual private responses of a large fraction of the US population (17–58% depending on the attacker's prior knowledge) [Abo21].

These results have proved to be controversial, however. Since this reconstruction is not "blatantly non-private", some researchers have argued that one could reconstruct a similarly large, *constant* fraction of the database by just picking attributes at random according to the population distribution (although under some rather *dubious* assumptions. . .) [Rug21]. Further research has since shown that reconstruction attacks significantly outperform random guessing, especially in small census blocks [DDK+23]. The possibility of such attacks has motivated the US Census bureau to adopt differential privacy for their data releases.

References

[Abo21] John M Abowd. Declaration of John M Abowd in case no. 3:21-cv-00211-rah-ecm-kcn, the State of Alabama v. United States Department of Commerce (2021). `https://www.documentcloud.org/documents/21018464-fair-lines-america-foundation-july-26-2021-declaration-of-john-m-abowd`, 2021.

[CN18] Aloni Cohen and Kobbi Nissim. Linear program reconstruction in practice. *arXiv preprint arXiv:1810.05692*, 2018.

[Coh22] Aloni Cohen. Attacks on deidentification's defenses. In *31st USENIX security symposium (USENIX Security 22)*, pages 1469–1486, 2022.

[DDK+23] Travis Dick, Cynthia Dwork, Michael Kearns, Terrance Liu, Aaron Roth, Giuseppe Vietri, and Zhiwei Steven Wu. Confidence-ranked reconstruction of census microdata from published statistics. *Proceedings of the National Academy of Sciences*, 120(8):e2218605120, 2023.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM Symposium on Principles of Database Systems*, pages 202–210, 2003.

[GAM19] Simson Garfinkel, John M Abowd, and Christian Martindale. Understanding database reconstruction attacks on public data. *Communications of the ACM*, 62(3):46–53, 2019.

[NS08] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE, 2008.

[Rug21] Steven Ruggles. Professor Steven Ruggles expert report in case no. 3:21-cv-00211-rah-ecm-kcn, the State of Alabama v. United States Department of Commerce (2021). `https://users.pop.umn.edu/~ruggles/censim/Ruggles_Report.pdf`, 2021.

[Swe00] Latanya Sweeney. Simple demographics often identify people uniquely. *Health (San Francisco)*, 671(2000):1–34, 2000.

[Swe02] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570, 2002.

[Swe15] Latanya Sweeney. Only you, your doctor, and many others may know. *Technology Science*, 2015092903(9):29, 2015.