Journal of Privacy and Confidentiality
Vol. 10 (1) 2020 TPDP 2018 Submitted Jan 2019
Published Jan 2020

LINEAR PROGRAM RECONSTRUCTION IN PRACTICE

ALONI COHEN AND KOBBI NISSIM

Boston University
e-mail address: aloni@bu.edu

Department of Computer Science, Georgetown University
e-mail address: kobbi.nissim@georgetown.edu

Abstract. We briefly report on a successful linear program reconstruction attack performed on a production statistical queries system and using a real dataset. The attack was deployed in test environment in the course of the Aircloak Challenge bug bounty program and is based on the reconstruction algorithm of Dwork, McSherry, and Talwar. We empirically evaluate the effectiveness of the algorithm and a related algorithm by Dinur and Nissim with various dataset sizes, error rates, and numbers of queries in a Gaussian noise setting.

Introduction

Responding to public and legislation pressures, companies are seeking practical and usable technological solutions to their data privacy problems. Larger corporations often employ Chief Privacy Officers and teams of privacy engineers to develop custom data privacy solutions. Some of these companies, including Google, Apple, and Uber, are recently experimenting with provable approaches to privacy using cryptography and differential privacy, which—at their current stage of development—require significant research and engineering efforts.

But not all companies have the means and technological sophistication to adopt this sort of bespoke approach to privacy. This void is being filled by a growing industry of companies selling off-the-shelf data privacy solutions, many of which aim to *anonymize* or *de-identify* sensitive data. These companies often advertise their anonymization products as not only preventing the disclosure of sensitive data but also ensuring compliance with relevant privacy

Key words and phrases: Data privacy, Reconstruction attacks.

Work done when the author was a student at MIT and partially when visiting Georgetown University. Supported by NSF Graduate Research Fellowship, Facebook Fellowship, NSF Project CNS-1413920. Code and data used in this work are available at `https://github.com/journalprivacyconfidentiality/Linear-Program-Reconstruction` and [Coh20]. More recent updates may be found at `https://github.com/a785236/Linear-Program-Reconstruction`.

www.journalprivacyconfidentiality.org
DOI:10.29012/jpc.711

laws, including HIPAA, FERPA, and GDPR.[1] Lack of transparency surrounds some of these technologies. Even when disclosed, the technical underpinnings of many privacy protection claims are heuristic and hence hard to evaluate.

Heuristic approaches to data privacy are not typically ruled out by data privacy regulations. Furthermore, these regulations are not typically interpreted to require a strong level of protection from data privacy technology. For example, the EU's General Data Protection Regulation limits its scope to those "means reasonably likely to be used" to re-identify data [Eur16]. A report from the UK's Information Commissioner's Office interprets similar language in earlier legislation as requiring protection against "motivated intruders"; alas, these intruders are assumed to lack both "any prior knowledge" and "specialist expertise" [Off12]. This policy approach allows practitioners to argue that data privacy technologies can be deployed even when they can be theoretically demonstrated to be vulnerable to attacks, as *purely theoretical* attacks plausibly fall outside the scope of the relevant regulations.

We believe that this is an unhealthy state of affairs. However, one can hope to affect the legal interpretation of existing regulations by implementing theoretical attacks and demonstrating their practicality. As a striking example, the decision to use differential privacy for the 2020 Decennial Census in the US was largely motivated by the Census Bureau's realization that traditional statistical disclosure limitation techniques may be vulnerable to practical reconstruction attacks [Abo18].

Reconstruction attacks. Academics have developed tools to reason formally about aspects of data privacy. One of these tools is reconstruction attacks, presented by Dinur and Nissim in 2003 [DN03]. They considered an adversary issuing count queries to a dataset $\mathbf{x}$ of n entries and showed that if the queries are answered with accuracy $o(\sqrt{n})$ then after making $\tilde{O}(n)$ such queries the adversary can reconstruct, by solving a simple linear program, all but a small fraction of the entries in $\mathbf{x}$. This result was further generalized and strengthened in [DMT07, DY08].

Reconstruction attacks have been mostly considered in the theoretical literature. They provide limits on the functionality provided privacy preserving mechanisms under *any* reasonable definition of privacy. Such analyses proved extremely useful in guiding the development of a mathematically rigorous reasoning about privacy, and the introduction of differential privacy in 2006 [DMNS06].

Reconstruction in practice. In this work, we apply a linear reconstruction attack on a statistical query system in the wild. To the best of our knowledge, this is the first time that such an attack has been successfully applied to reconstruct data from a commercially-available statistical database system specifically designed to protect the privacy of the underlying data.

We performed the attack on a production system called Diffix on a real dataset.[2] The attack was deployed in a test environment in the course of a bug bounty program by Aircloak, the makers of Diffix. The goal of Diffix is to allow data analysts to perform an unlimited

[1] HIPAA is the US Health Insurance Portability and Accountability Act. FERPA is the US Family Educational Rights and Privacy Act. The GDPR is the EU General Data Protection Regulation.

[2] By "Diffix" we refer to the system as described in [FEO+18] and which was available for the Aircloak bug bounty program. To the best of the authors' knowledge, this refers to the version now called "Diffix-Birch".

number of statistical queries on a sensitive database while protecting the underlying data and while introducing only minimal error. It is advertised as an off-the-shelf, GDPR-compliant privacy solution, and Aircloak reports that "CNIL, the French national data protection authority, has already evaluated Diffix against the GDPR anonymity criteria, and have stated that Aircloak delivers GDPR-level anonymity" [AIR18b].

As we show, by answering unlimited, highly accurate statistical queries Diffix is vulnerable to linear reconstruction attacks.

1. Diffix

Diffix is a system that sits between a data analyst and a dataset. The data analyst issues count queries using a restricted subset of SQL. For example,

```
SELECT count(*) FROM loans
WHERE status = 'C'
AND client-id BETWEEN 2000 and 3000
```

Diffix executes a related query on the underlying dataset and then returns the answer to the analyst's query masked with additional error.

A primary focus of Diffix's design is noise generation. The answer to every query is masked with error generated by adding multiple samples from zero-mean Gaussian distributions and, depending on the data, rounding to the nearest integer. The standard deviation of the error grows with the square root of the number of conditions in the SQL query. (The above query has two conditions: `status = 'C'` and `client-id BETWEEN 2000 and 3000`.) To generate this noise, samples from $N(0, 1)$ are generated for every condition in the SQL query, are added together with a baseline error, and are used to mask the final answer.

That the standard deviation scales with the square root of the number of conditions was thought to prevent linear reconstruction attacks [FEO+18] (we explain that intuition in Section 3). The precise structure of the noise—designed to thwart specific families of attacks—is not otherwise relevant to this work. For instance, our attack ignores the fact that queries with overlapping conditions have correlated noise. The text of the conditions are used to seed the pseudo-random number generator used to sample the Gaussians, and identical conditions yield identical seeds.

In addition to adding Gaussian noise, Diffix employs a number of heuristic techniques. To protect against an attacker who may try to average noise out by issuing many logically equivalent but syntactically distinct queries, Diffix restricts the use of certain SQL operators, especially math operators. These restrictions ultimately dictated the peculiar form of our attack queries. Other techniques include suppressing small counts, modifying extreme values, and disallowing many SQL operators (e.g., `OR`).

In order to accelerate development and testing of our linear reconstruction attack, we simulated Diffix's noise addition and small-count suppression in MATLAB. The results in Section 3 use real query responses from Diffix, while those in Section 4 use the simulation.

The Challenge. From December 2017 to May 2018, Aircloak ran "the first bounty program for anonymized data re-identification," offering prizes of up to $5,000 for successful attacks [AIR18a]. We commend Aircloak for making Diffix available to privacy researchers and for their support throughout this work.

Aircloak granted researchers API access to 5 datasets through Diffix, along with documentation of Diffix's design and implementation and complete versions of the datasets for analysis. Researchers were allowed to use auxiliary information gleaned directly from the datasets in order to carry out their attacks. Our target was a dataset of banking data from the Czech Republic, originally released as part of a data challenge for the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD '99).[3]

Aircloak measured the success of an attack using an effectiveness parameter α and a confidence improvement parameter κ. They have verified our attack to achieve the best possible parameters. In a recent blog post, Aircloak reported that "Only two attack teams formulated successful attacks. ... Fixes for both attacks have been implemented" [AIR18c]. We have not examined whether the new restrictions on the query language introduced by Aircloak counter linear reconstruction attacks.

2. Linear Reconstruction

Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a database of binary values $x_i \in \{0, 1\}$. A subset-sum query identifies a subset q of $\{1, \ldots, n\}$, and the answer to the subset-sum query is $q(x) = \sum_{i \in q} x_i$, an integral number in the range 0 to n. Consider a mechanism M with access to the database $\mathbf{x}$ which answers subset sum queries. In an attempt to protect the privacy of the underlying data, the mechanism M may perturb its answers. Namely, given a query q, it returns an answer a_q which may differ from the exact answer $q(x)$. We say that M has error E if for all queries q,

$$a_q = q(x) + e_q \text{ where } |e_q| \leq E.$$

Dinur and Nissim defined reconstruction attacks as those where an adversary A with query access to M manages (with high probability) to learn an approximation $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n) \in \{0, 1\}^n$ for $\mathbf{x}$ within sublinear Hamming distance, i.e.,

$$|\{i : \hat{x}_i \neq x_i\}| = o(n). \tag{2.1}$$

They observed that with each query q made by an adversary A to the database $\mathbf{x}$ via the mechanism M the adversary A obtains a linear threshold constraint on the unknown $(x_1, \ldots, x_n)$, namely:

$$-E \leq \sum_{i \in q} x_i - a_q \leq E.$$

These linear constraints (plus the restriction of each x_i to the range $[0, 1]$) form a linear program, for which the underlying dataset $\mathbf{x}$ is a feasible solution.

Let $E = o(\sqrt{n})$ and consider the linear program consisting of the constraints obtained by making $\tilde{O}(n)$ randomly chosen queries. By solving this linear program and then rounding the result to $\{0, 1\}$ the adversary can reconstruct an approximation $\hat{\mathbf{x}}$ satisfying Eq. 2.1. In other words, if M allows making $\tilde{O}(n)$ arbitrary subset-sum queries, which it answers with error $E = o(\sqrt{n})$, then M is susceptible to a reconstruction attack, and hence M does not preserve privacy under any reasonable definition of privacy. This linear reconstruction attack was further generalized and strengthened in a sequence of works [DMT07, DY08, KN13]. In particular, Dwork McSherry and Talwar [DMT07] removed the requirement for all answers

[3]Dataset and description available at https://sorry.vse.cz/~berka/challenge/PAST/. See also https://aircloak.com/wp-content/uploads/Aircloak_Challenge.pdf.

of M to be within error E, and the attack we describe in this work is a simple variant of their attack.

3. Implementing the Linear Reconstruction Attack

Setup. The attack targets a dataset $\mathbf{x}$ of size n database entries indexed by a set of unique identifiers $\mathcal{I}$. Each entry has an associated value of a Boolean target attribute, x_i. Each query $q \subseteq [n]$ specifies a subset of entries, and the response $a_q = q(\mathbf{x}) + e_q$ is the sum of true value $q(\mathbf{x}) = \sum_{i \in q} x_i$ and an error term e_q. The errors are sampled from a zero-mean Gaussian distribution of standard deviation σ, then rounded to the nearest integer.

We implemented a linear reconstruction attack following the approach of [DMT07] to find a candidate database $\mathbf{x}'$ minimizing the total error. [DMT07] was designed for a setting when some errors may be very significant, but typical errors are small. In contrast, the linear program of [DN03] from the previous section is suitable when there is a bound on the maximum error magnitude. Although we use the linear program of [DMT07], we deviate by using subset queries. That work analyzes a number of other types of queries, including ± 1 queries of the form $q^{\pm}(\mathbf{x}) = \sum_{i \in q} x_i - \sum_{i \notin q} x_i$ for subsets $q \subseteq [n]$. While these queries can be implemented using subset queries,[4] the standard deviation of the resulting noise would be larger. In contrast, subset queries were directly implementable in Diffix with less noise and proved effective. Section 4 reports on additional experiments testing the accuracy of these three contrasting approaches in the face of Gaussian noise.

We solve the following linear program over $n + m$ variables $\mathbf{x}' = (x'_i)_{i \in \mathcal{I}}$ and $(e'_q)_{q \in Q}$:

$$
\begin{aligned}
&\textbf{variables: } \mathbf{x}' = (x'_i)_{i \in \mathcal{I}} \text{ and } (e'_q)_{q \in Q} \\
&\textbf{minimize: } \textstyle\sum_{q \in Q} |e'_q| \\
&\textbf{subject to:} \\
&\qquad \forall q \in Q, \quad e'_q = a_q - q(\mathbf{x}') \\
&\qquad \forall i \in \mathcal{I}, \quad 0 \le x'_i \le 1
\end{aligned}
$$

There is a standard linearizing of the above nonlinear objective function by introducing m additional variables. To compute the final output, we round the real-valued x'_i to the nearest value in $\{0, 1\}$.

The results described in this section are from a reanalysis of data gathered during the Aircloak Challenge using the linear program described above. During the course of the actual challenge, we used a slightly modified linear program as described in Appendix A.

Querying "random" subsets. The main hurdle in implementing the attack was specifying queries for random subsets of the rows of the dataset.

The attacks in [DN03] and [DMT07] make use of queries which are random subsets of $[n]$. The most straightforward way to specify subsets $q \subseteq [n]$ in a SQL query is to use $|q| = \Theta(n)$ conditions, one for each $i \in q$. If q is selected at random, it is impossible to do significantly better because q is not compressible. But recall that Diffix determines the error magnitude per query depending on the description of the query. The standard deviation hence grows with the square root of the number of conditions. For a random subset q,

[4] $q^{\pm}(\mathbf{x}) = q(\mathbf{x}) - q^c(\mathbf{x})$.

the noise hence grows as $\sqrt{n}$, too large for linear program reconstruction to work. Thus, [FEO+18] argued that Diffix thwarts linear program reconstruction.

To reconstruct the dataset, we needed to find a way to specify a random—or "random enough"—subset of the data using as few conditions as possible. Our approach, ad hoc yet ultimately effective, was to use the unique user identifier i as the source of "randomness."

Each query was specified by a prime p, an offset j, an exponent $e \in \{0.5, 0.6, \ldots, 1.9\} \setminus \{1\}$, and a test $\phi \in \{\phi_2, \phi_5\}$. The query $q = (\phi, p, j, e)$ is the set $\{i \in [n] : \phi(i; p, j, e) = 1\}$, where

$\phi_2(i; p, j, e) = 1$ iff the jth digit in the decimal representation of $(p \cdot i)^e$ is even, and

$\phi_5(i; p, j, e) = 1$ iff the jth digit in the decimal representation of $(p \cdot i)^e$ is less than 5.

For example, the following query corresponds to $p = 2$, $j = 2$, $e = 0.7$ and ϕ_5. To implement ϕ_2 replace each 100 with 500.

```
SELECT count(clientId) FROM loans
WHERE floor(100 * ((clientId * 2)^0.7) + 0.5)
    = floor(100 * ((clientId * 2)^0.7))
```

The exact form of the query depended on the various syntactic restrictions included in Diffix. By modifying the ranges of p and j, we were able to tune the total number of queries. We restricted p to the first 25 primes and $j \in [5]$, resulting in a total of 3500 queries.

Results. Our target was the loans table in the banking dataset, consisting of real data of 827 loans from a bank in the Czech Republic. The rows are indexed by the clientId attribute, a unique number between 2 and 13971. Each row has an associated loanStatus attribute, a letter from 'A' to 'D'.

Our goal was to determine which loans had loanStatus = 'C', given only knowledge of the clientIds. In order to minimize the total number of queries, we restricted our attention to the subset of clientIds in the range $[2000, 3000]$, which contained 73 entries. Ultimately, our queries were of the form:

```
SELECT count(clientId) FROM loans
WHERE floor(100 * ((clientId * 2)^0.7) + 0.5)
    = floor(100 * ((clientId * 2)^0.7))
AND clientId BETWEEN 2000 and 3000
AND loanStatus = 'C'
```

The Diffix API reported it added error of standard deviation 4 to the output of these queries. We applied the same attack on different ranges of clientIds with 110, 130, and 142 entries (and in the last case, targeting the loanStatus value 'A'). In each case, we performed 3500 queries.

The linear program reconstructed the data for all four clientId ranges perfectly.

Discussion. Can Diffix be adapted to prevent linear program reconstruction? If enough counting queries over subsets of a dataset are answered with error bounded by $o(\sqrt{n})$, then the linear program from [DN03, DMT07] reconstructs all but a small fraction of the underlying dataset. Diffix could of course prevent the attack by increasing the magnitude of the noise or limiting the number of queries an analyst can make. However, [FEO+18] describes Diffix as "minimiz[ing] the amount of noise needed to strongly protect the anonymity of individuals in the database, and eliminat[ing] the need for the budget that is found in systems based on

differential privacy." Modifying Diffix in this way would be a change to the basic goals of the system. Instead, Aircloak has chosen to further restrict the types of queries an analyst can make. Though we have not examined the changes introduced by Aircloak to counter our attack, it is our understanding that newer versions of Diffix would not answer the specific queries used by our attack.

4. Simulated Experiments

The above attack leaves a number of unanswered questions. How does the effectiveness of the attack vary with the magnitude of the Gaussian error, the number of queries, the database size, or the reconstruction algorithm? Is prior knowledge of the clientIds necessary?

We explore these questions using a simulation of Diffix's noisy statistical query mechanism. The simulated mechanism answers count queries with zero-mean, normally-distributed noise with standard deviation σ (and rounds to the nearest integer). It also suppresses low counts in the same way as the Diffix system, though that was only be relevant for the first experiment. All experiments described below were implemented in MATLAB on a personal laptop, and all linear programs were solved in less than 4 seconds.

Removing auxiliary information. One drawback of our original attack on Diffix was the need for complete knowledge of the clientIds as a prerequisite to performing the attack. Our first experiment sought to infer these clientIds. First, we identified a range of 100 possible clientIds that had a large number of present clientIds (relative to the other possible ranges). We want a large number of present clientIds to minimize the effect of Diffix's low-count suppression. We settled on the range $[2500, 2600]$ with 12 clientIds. While we identified this range using exact counts, we believe such a range could be found by querying Diffix itself.[5]

We simulated responses to 3500 queries of the following form:

```
SELECT count(clientId) FROM loans
WHERE floor(100 * ((clientId * 2)^0.7) + 0.5)
    = floor(100 * ((clientId * 2)^0.7))
AND clientId BETWEEN 2500 and 2600
```

The [DMT07] linear program was used to infer which clientIds are present in the range. There was 1 false negative among the 12 present clientIds and 0 false positives among the 88 absent clientIds.

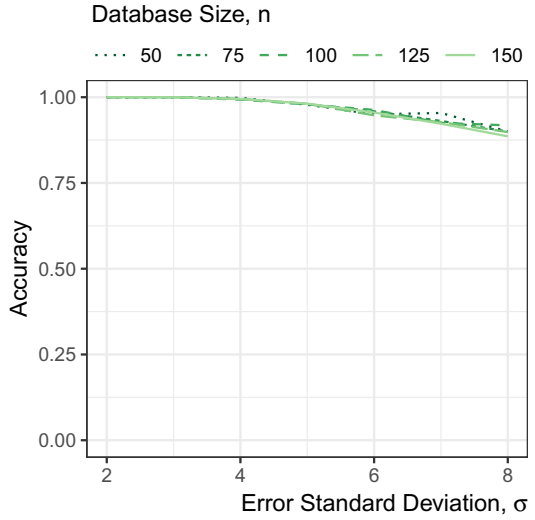
How accuracy varies with size, queries, and error. The accuracy of the linear reconstruction attack depends on the size of the dataset, the magnitude of the error, and the number of queries. When implementing our attack on Diffix, we used many more queries than seemed necessary for the level of noise used. The next experiment illustrates how the accuracy of [DMT07] varies with each of these parameters against a system using Gaussian noise to answer count queries.

The results are summarized in Figure 1. The plots display the average accuracy over 10 simulated runs of our [DMT07] reconstruction algorithm as the error magnitude, database size, and number of queries were varied. Each run resampled the Gaussian noise while the

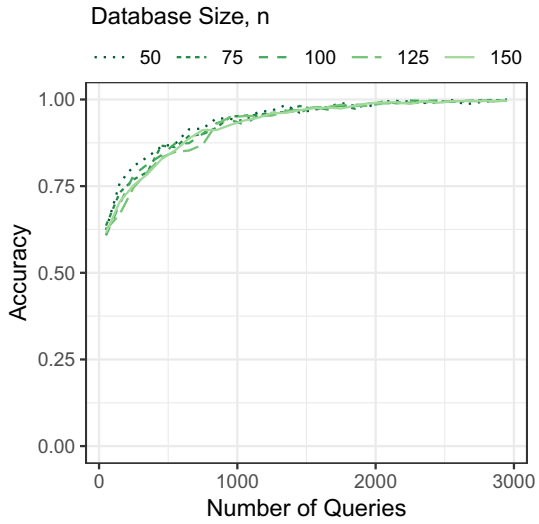
[5] *E.g.*, by issuing the query `SELECT count(*) FROM loans WHERE clientid BETWEEN` a `and` b to approximate the number of present clientIds in the range $\{a, \ldots, b\}$.

underlying dataset remained fixed. It is interesting to observe that the size of the dataset does not seem to significantly affect the effectiveness of reconstruction.

As described in Section 3, the analysis in [DMT07] applies to ± 1 queries but not to subset queries. To compare the effectiveness of these two query types, we ran the same simulations using ± 1 queries. The results are summarized in Figure 2. The plots are nearly indistinguishable from the corresponding plots in Figure 1.



(A) Noise standard deviation varied from 1 to 8, in increments of 1, with 2550 queries. For $n = 100$, the mean accuracy falls below 0.99 at $\sigma = 5$ and below 0.95 at $\sigma = 7$.



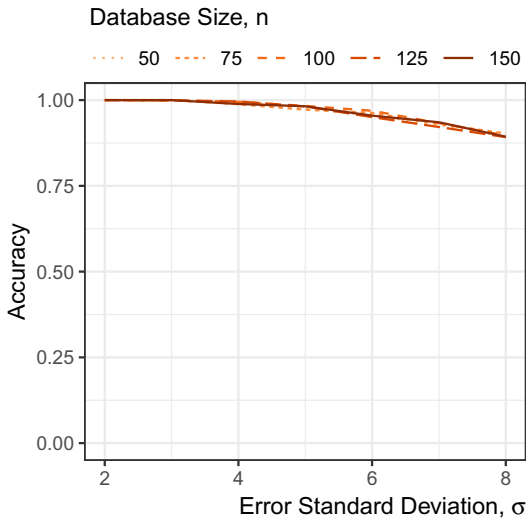
(B) The number of queries varied from 50 to 2950, in increments of 100, with noise magnitude $\sigma = 4$. For $n = 100$, the mean accuracy surpasses 0.95 at 1150 queries and surpasses 0.99 at 2050 queries.

FIGURE 1. Reconstruction accuracy as a function of the (1a) noise magnitude and (1b) number of queries, for various database sizes. The data is averaged over 10 trials of the [DMT07] linear program using subset queries.

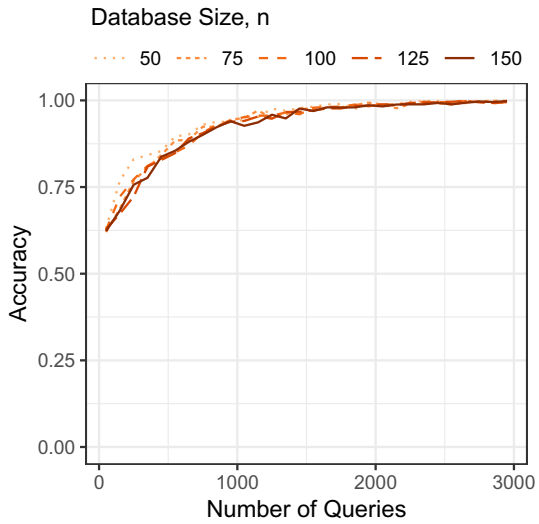
Comparing [DN03] and [DMT07]. The original linear reconstruction attack for noisy statistical queries comes from [DN03]. In contrast to [DMT07], [DN03] makes the additional assumption that each error e_q is bounded by a maximum error $\mathcal{E}$. In our experiments, we write $\mathcal{E} = B\sigma$, where B is the *error bound multiplier* and σ is the standard deviation of the Gaussian errors. The [DN03] linear program reflects the bounded-error assumption with an additional constraint and uses a trivial objective function.

variables: $\mathbf{x}' = (x'_i)_{i \in \mathcal{I}}$ and $(e'_q)_{q \in Q}$
minimize: 0
subject to:

$$\begin{aligned} \forall q \in Q, \quad & e'_q = a_q - q(\mathbf{x}') \\ & e'_q \leq B\sigma \\ \forall i \in \mathcal{I}, \quad & 0 \leq x'_i \leq 1 \end{aligned}$$



(A) Noise standard deviation varied from 1 to 8, in increments of 1, with 2550 queries. For $n = 100$, the mean accuracy falls below 0.99 at $\sigma = 5$ and below 0.95 at $\sigma = 8$.



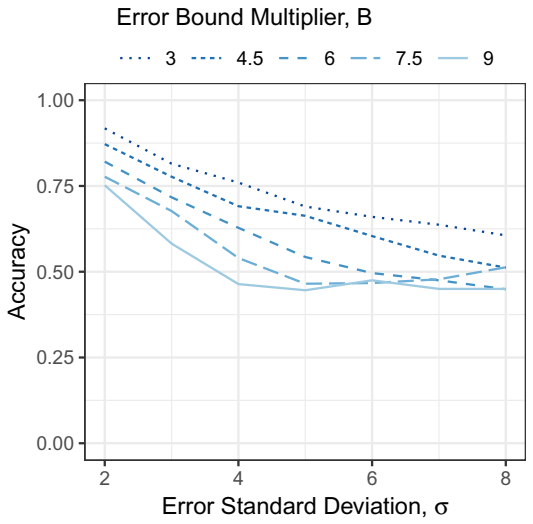
(B) The number of queries varied from 50 to 2950, in increments of 100, with noise magnitude $\sigma = 4$. For $n = 100$, the mean accuracy surpasses 0.95 at 1050 queries and surpasses 0.99 at 2050 queries.

FIGURE 2. Reconstruction accuracy as a function of the (2a) noise magnitude and (2b) number of queries, for various database sizes. The data is averaged over 10 trials of the [DMT07] linear program using ± 1 queries.

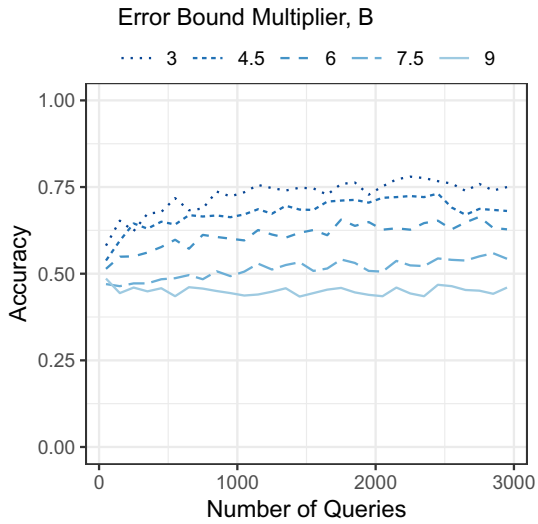
Our final experiment illustrates how the accuracy of the above [DN03]-based linear program varies as a function of the error magnitude, number of queries, and the error bound multiplier. The results are summarized in Figure 3. The plots display the average accuracy over 10 simulated runs of our [DN03] reconstruction algorithm as the parameters were varied. Each run resampled the Gaussian noise while the underlying dataset remained fixed.

Observe that as the error bound multiplier B increases, the accuracy of reconstruction degrades. Because the linear program terminates once any feasible point is found, it is not surprising that expanding the set of feasible points (by increasing B) hurts accuracy.

Note however that the pattern extends to $B = 3$. One would expect a few queries (in expectation about 4.6 queries per 2550 for $\sigma = 4$) to have rounded error greater than 3σ. Nevertheless, in each of the 240 trials run with $B = 3$ and at least 2550 queries, a feasible solution was found. In contrast, for $B = 2.5$ and $\sigma = 4$ half of all executions with 1850 queries were infeasible (dropping to $\geq 90\%$ infeasible at 2250 or more queries).



(A) Noise standard deviation varied from 1 to 8, in increments of 1, with 2550 queries and dataset size $n = 100$.



(B) The number of queries varied from 50 to 2950, in increments of 100, with noise magnitude $\sigma = 4$ and dataset size $n = 100$.

FIGURE 3. Accuracy as a function of the (3a) noise magnitude and (3b) number of queries, for various values of the DiNi multiplier B. The data is averaged over 10 trials using a dataset of size $n = 100$.

References

[Abo18] John Abowd. Staring-down the database reconstruction theorem, July 2018. https://www.census.gov/content/dam/Census/newsroom/press-kits/2018/jsm/jsm-presentation-database-reconstruction.pdf.

[AIR18a] Aircloak attack challenge, 2018. https://aircloak.com/solutions/attack-challenge-en/.

[AIR18b] Data anonymisation: What it is and why it matters. 2018. https://aircloak.com/wp-content/uploads/Data-Anonymisation-What-it-is-and-Why-it-Matters.pdf.

[AIR18c] Data compliance in the gdpr - how anonymisation allows you to stay compliant in your data analysis, Aug 2018. https://aircloak.com/data-compliance-in-the-gdpr/.

[Coh20] Aloni Cohen. Code and data for Linear Program Reconstruction in Practice, January 2020. NSF Graduate Research Fellowship, Facebook Fellowship, NSF Project CNS-1413920. 10.5281/zenodo.3608315.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006. 10.1007/11681878_14.

[DMT07] Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of LP decoding. In *STOC*, pages 85–94. ACM, 2007. 10.1145/1250790.1250804.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210. ACM, 2003. 10.1145/773153.773173.

[DY08] Cynthia Dwork and Sergey Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 469–480. Springer, 2008. 10.1007/978-3-540-85174-5_26.

[Eur16] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, L119:1–88, May 2016. http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC.

[FEO+18] Paul Francis, Sebastian Probst Eide, Pawel Obrok, Cristian Berneanu, Sasa Juric, and Reinhard Munz. Extended diffix. *CoRR*, abs/1806.02075, 2018. http://arxiv.org/abs/1806.02075, arXiv:1806.02075.

[KN13] Alexander Kantor and Kobbi Nissim. Attacks on statistical databases: The highly noisy case. *Inf. Process. Lett.*, 113(12):409–413, 2013. 10.1016/j.ipl.2013.03.005.

[Off12] UK Information Commissioner's Office. *Anonymisation: managing data protection risk code of practice*. Nov 2012. https://ico.org.uk/media/1061/anonymisation-code.pdf.

Appendix A. Additional Information on the Aircloak Challenge Attack

The results described in Section 3 are from a reanalysis of data gathered during the Aircloak Challenge. During the course of the Aircloak Challenge, we used a modified version of the [DMT07] linear program. For transparency, this section describes the modified linear program and its effectiveness.

The only difference between the linear program originally used and the one described in Section 3 is the addition of constraints upper bounding the magnitude of any error term.

$$\begin{aligned}
&\textbf{variables: } \mathbf{x}' = (x'_i)_{i\in\mathcal{I}} \text{ and } (e'_q)_{q\in Q}\\
&\textbf{minimize: } \textstyle\sum_{q\in Q} |e'_q|\\
&\textbf{subject to:}\\
&\qquad \forall q \in Q, \quad e'_q = a_q - q(\mathbf{x}')\\
&\qquad\qquad\quad\;\; e'_q \le 5\sigma\\
&\qquad \forall i \in \mathcal{I}, \quad 0 \le x'_i \le 1
\end{aligned}$$

where $\sigma = 4$ is the standard deviation of the true error distribution. Note that if the true errors were distributed according to $N(0, \sigma)$ and rounded to the nearest integer, an error of magnitude greater than 5σ would be expected once in every 1.7 million queries.

We first implemented the linear reconstruction solver using data from the clientId range $[2000, 3000]$, for which it achieved perfect reconstruction. Together with researchers at the Max Planck Institute for Software Systems, we verified the attack on three additional ranges of clientIds containing 110, 130, and 142 clientIds. The results are summarized in Table 1. In two of the three ranges, the attack again inferred whether each loanStatus was 'C' with high accuracy (1 and .9538). We were surprised, therefore when the final validation (this time targeting loanStatus 'A' rather than 'C') achieved accuracy of only 75.4%. Our confusion compounded when the accuracy degraded after increasing the number of queries, suggesting that we were not accounting for some source of error.

After further investigation, we realized that performing the queries for clientIds in $[10000, 12000]$ required more numerical precision than seemed to be supported by Diffix. The larger clientId values in this range and the larger constants required to answer additional queries introduced errors that had not affected our earlier tests. Ultimately, high accuracy was recovered by ignoring the results from queries with larger values of e (which require greater precision), but making no other changes to the linear program solver.

clientIds Range	Number of entries (n)	Number of queries	Target status	Accuracy
2000-3000	73	3500	'C'	1
3000-5000	110	3500	'C'	1
5000-7000	130	3500	'C'	.9538
10000-12000	142	3500	'A'	.7535
10000-12000	142	2000, $e \le 1.4$	'A'	1
10000-12000	142	1000, $e \le 0.8$	'A'	.9930

Table 1. Summary of reconstruction tests performed against Diffix using the modified [DMT07] linear program. The queries in the final two resulted from restricting the exponent e to the indicated range.