When the Signal is in the Noise: Exploiting Diffix's Sticky Noise

Andrea Gadotti[*a], Florimond Houssiau[*a], Luc Rocher[*a,b], Benjamin Livshits[a], and Yves-Alexandre de Montjoye[†a]

[a]*Department of Computing and Data Science Institute, Imperial College London*
[b]*ICTEAM, Université catholique de Louvain*

Abstract

Anonymized data is highly valuable to both businesses and researchers. A large body of research has however shown the strong limits of the de-identification release-and-forget model, where data is anonymized and shared. This has led to the development of privacy-preserving query-based systems. Based on the idea of "sticky noise", Diffix has been recently proposed as a novel query-based mechanism satisfying alone the EU Article 29 Working Party's definition of anonymization. According to its authors, Diffix adds less noise to answers than solutions based on differential privacy while allowing for an unlimited number of queries.

This paper presents a new class of noise-exploitation attacks, exploiting the noise added by the system to infer private information about individuals in the dataset. Our first differential attack uses samples extracted from Diffix in a likelihood ratio test to discriminate between two probability distributions. We show that using this attack against a synthetic best-case dataset allows us to infer private information with 89.4% accuracy using only 5 attributes. Our second cloning attack uses dummy conditions that conditionally strongly affect the output of the query depending on the value of the private attribute. Using this attack on four real-world datasets, we show that we can infer private attributes of at least 93% of the users in the dataset with accuracy between 93.3% and 97.1%, issuing a median of 304 queries per user. We show how to optimize this attack, targeting 55.4% of the users and achieving 91.7% accuracy, using a maximum of only 32 queries per user.

Our attacks demonstrate that adding data-dependent noise, as done by Diffix, is not sufficient to prevent inference of private attributes. We furthermore argue that Diffix alone fails to satisfy Art. 29 WP's definition of anonymization. We conclude by discussing how non-provable privacy-preserving systems can be combined with fundamental security principles such as defense-in-depth and auditability to build practically useful anonymization systems without relying on differential privacy.

[†]Email: deMontjoye@imperial.ac.uk; Corresponding author.

We acknowledge the support from Agence Française de Développement. The opinions expressed in this article are the authors' own and do not reflect the view of the Agence Française de Développement.
Luc Rocher is the recipient of a doctoral fellowship from the Belgian National Fund for Scientific Research (F.R.S.-FNRS).

1 Introduction

Personal data holds a significant potential for researchers and organizations alike, yet its large-scale collection and use raise serious privacy concerns. While scientists have compared the impact of modern large-scale datasets of human behaviors to the invention of the microscope [1], numerous scandals, such as the recent Cambridge Analytica debacle [2] highlight the importance of privacy and data protection for the general public and modern societies.

Historically, the balance between using personal data and preserving people's privacy has relied, both practically and legally, on the concept of data anonymization. Anonymization, also called de-identification, is the process of transforming personal data to mask the identity of participants, e.g. by removing identifiers, coarsening data, or adding noise. The recent European General Data Protection Regulation (GDPR) defines anonymous information as "information which does not relate to an identified or identifiable natural person or to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable" [3]. Similar definitions exist in other protection laws around the world, such as the HIPAA privacy rule for medical data in the US and the ePrivacy regulation. These all state that anonymized data does not require consent from participants to be shared widely, as it cannot be traced back and potentially used against them.

While de-identification algorithms are widely used in industry and academia to transform and release anonymous datasets, a large body of research has shown that these practices are not resistant to a wide range of re-identification attacks [4–9]. Exposure of the limits of de-identification have led to less than happy conclusions by policy makers: for instance, the [US] President's Council of Advisors on Sci-

ence and Technology concluded that data anonymization "is not robust against near-term future re-identification methods. PCAST do not see it as being a useful basis for policy" [10].

Query-based systems. In response to the limits of de-identification, privacy researchers and companies have proposed query-based systems as an alternative. Such systems typically offer data analysts a remote interface to ask questions that return data aggregated from several, potentially many, records. Granting access to the data only through queries, without releasing the underlying raw data, mitigates the risk of typical re-identification attacks. Yet a malicious analyst can often submit a series of seemingly innocuous queries whose outputs, when combined, will allow them to infer private information about participants in the dataset.

Differential privacy. Privacy research has been increasingly focused on providing provable privacy guarantees to defend query-based systems against such attacks. Differential privacy [11] is the main privacy guarantee considered by researchers. Intuitively, a randomized algorithm is differentially private if the output does not depend on any single individual's record in the dataset. It has been shown that algorithms that satisfy differential privacy are robust to a very large class of attacks [12]. Yet, efficient differential privacy mechanisms are generally very use case-specific and, even if a large range of differentially private mechanisms have been proposed, there is still no practical widely deployed differential privacy solution for general-purpose data analytics [13]. To achieve strong privacy guarantees, practitioners must often sharply limit the data utility by adding large amounts of noise and restrict the total number of requests that the system is allowed to answer [14]. Moreover, while differential privacy is a strong guarantee, the risk of data breaches because of implementation issues remains, exposing systems to attacks that differential privacy should in theory rule out [15, 16]. Overall, with some rare exceptions, the complexity of correctly implementing differential privacy and choosing the right privacy budget often prevents practitioners from using it.

Alternatives to differential privacy. Diffix, a patented commercial solution that acts as an SQL proxy between an analyst and a protected database [17, 18], has recently been proposed by researchers affiliated with Aircloak and the Max Planck Institute for Software Systems as a practical alternative to differential privacy, based on the [EU] Article 29 Working Party (Art. 29 WP)'s definition of anonymous data. It defines a dataset as anonymous if the anonymization mechanism used protects against *singling out* (identify one user), *linkability* (match users across datasets), and *inference* (learn participants' records) attacks [18, 19]. Diffix relies on a novel noise addition framework called "sticky noise", which aims to give analysts a rich query syntax, minimal noise addition, and an infinite number of queries, all while satisfying the WP29 definition of anonymous.

The authors claim that data produced by Diffix (*i*) falls outside of the scope of the new European GDPR regulation; (*ii*) has been determined by the French National Commission on Informatics and Liberty (CNIL) to offer "GDPR-level anonymization" for all cases; and (*iii*) has been certified by TÜViT as fulfilling "all requirements for data collection and anonymized reporting" [20, 21]. Diffix is used in production and Aircloak reports working with partners such as Telefonica, DZ Bank and Cisco.

Exploiting Diffix's noise. In this paper we present a new class of attacks, called noise-exploitation attacks. The attacks work in three parts: (*i*) canceling out part of the sticky noise using multiple queries, (*ii*) exploiting the noise Diffix adds to one query in order to learn information about the query set associated to this query, and (*iii*) using logical equivalence between queries to obtain independent noise samples for the same query. We develop two noise-exploitation attacks that take advantage of the structure of Diffix's sticky noise to infer private (also called secret) attributes of individuals in the dataset, violating the inference requirement from the Article 29 WP definition of anonymization [19]. Our first attack, the differential attack, uses samples obtained by the difference between two queries' outputs, to discriminate between two distributions, depending on the value of the private attribute. We show that, under specific conditions, this attack potentially allows an attacker to infer private information of unique users with up to 96.8% accuracy knowing only 5 pieces of auxiliary information we call attributes.

Our second noise-exploitation attack, the cloning attack, uses dummy conditions that affect the output of queries conditionally to the value of the private attribute. This attack relies on weaker assumptions and automatically validates them with high accuracy, without the need for an oracle. It proceeds in two steps: (*i*) a *validation* step, searching for subsets of known attributes to use for the attack, that will satisfy the assumptions required for its success, and (*ii*) an *inference* step that uses the attributes found to predict users' private attribute's value. We perform the attack against four real-world datasets, and show that it can infer the private attributes of between 87.0% and 97.0% of all records across datasets. We then present an optimized cloning attack that targets 55.4% of the users and achieves the same accuracy using as little as 32 queries. This proves that introducing limits for the number of allowed queries would not protect against our attacks.

Contributions. This paper makes the following contributions:

- By developing and implementing two attacks, we demonstrate that Diffix alone does not currently satisfy the *inference* requirements of the Art. 29 WP. We make the datasets and code for the attacks and experiments available to other researchers.
- We show, using a collection of four previously published datasets that the assumptions made by our attacks are realistic. We establish that, across all datasets, between

93% and 100% of all users are value-unique (i.e. all records sharing the same set of attributes have the same private attribute).
- We make a range of defense-in-depth proposals, which can be used to improve the practical privacy guarantees of both Diffix and other non-differentially private data anonymization tools. While these measures will not result in differential privacy guarantees, they might provide adequate practical solution in many settings.
- We show, using the Diffix mechanism as our primary example, that anonymization mechanisms that do not rely on differential privacy might not be GDPR-compliant alone, and that naive data-dependent noise can lead to powerful attacks.

Paper organization. The rest of the paper is organized as follows. Section 2 describes the Diffix mechanism. Section 3 presents two attacks exploiting the noise added by the system to infer private attributes of individuals in the dataset. Section 4 shows, on real-world datasets, how an attacker can accurately attack the Diffix mechanism. Section 5 discusses the assumptions of the attacks and potential solutions for Diffix to thwart noise-exploitation attacks moving forward. Sections 6 and 7 summarize related work and provide our conclusions to build practically useful anonymization systems. Appendix A provides some details for the statistical tests used by the attack. Appendix B describes how to optimize the cloning attack to reduce the number of queries.

2 Summary of the Diffix framework

Here we summarize the Diffix framework as described in [18] and introduce notations for our attack. Diffix acts as an SQL proxy between an analyst and a *protected database D* where each row is an individual record and each column one attribute. The analyst can send SQL queries to Diffix, which will process the queries and then output a noisy answer.

We denote with $\mathcal{A}_D$ the set of attributes in the database D. For instance, $\mathcal{A}_D$ could contain 4 attributes $\mathcal{A}_D = \{gender, age, zip, HIV\}$ with HIV a binary attribute (0 or 1). A record x is a row of D with values for the attributes in $\mathcal{A}_D$. For example, with $\mathcal{A}_D$ as above, we could have $x = (M, 27, 55416, 1)$. We assume, for simplicity, that there is one and only one record for every user in D.

While Diffix can process a large part of the SQL syntax, we here focus on simple count queries:

SELECT count(∗)
FROM *table*
WHERE $condition_1$ AND $condition_2$ [AND ...]

where every condition is an expression of the form "$attribute \square value$" with $\square$ being $=, \neq, \leq, <, \geq$, or $>$.

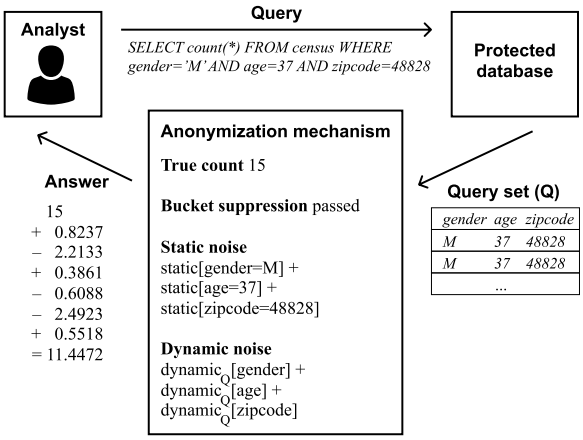

Figure 1: Diffix privacy-preserving architecture

For simplicity, we use a shorter notation for queries using "∧" for the logical AND:

$$Q \equiv \mathsf{count}(condition_1 \wedge condition_2 \wedge \ldots).$$

The following query would, for example, count the number of individuals in the database who are male, 37 years old, and live in the area with ZIP code 48828:

$$Q \equiv \mathsf{count}(gender = M \wedge age = 37 \wedge zip = 48828)$$

Diffix's privacy-preserving mechanism. Diffix protects privacy through *sticky noise* addition (static and dynamic noise) and bucket suppression (see Fig. 1). Let $Q \equiv \mathsf{count}(C_1 \wedge \ldots \wedge C_h)$, and denote by $Q(D)$ the true result of Q evaluated on D (without noise). Diffix's output for Q on D (without bucket suppression, see below) is:

$$\widetilde{Q}(D) = Q(D) + \sum_{i=1}^{h} \mathsf{static}[C_i] + \sum_{i=1}^{h} \mathsf{dynamic}_Q[C_i] \quad (1)$$

with $\mathsf{static}[C_i]$ the *static noise* for condition C_i, $\mathsf{dynamic}_Q[C_i]$ the *dynamic noise* for condition C_i in Q.

Static noise. Let C be a condition, for example $age = 34$. The *static noise* $\mathsf{static}[C]$ associated to C is a random number drawn from a normal distribution $\mathcal{N}(0,1)$. The value is generated by a pseudo-random number generator (PRNG), whose seed is a salted hash of the string literal C:

$$static\_seed_C = \mathsf{XOR}(\mathsf{hash}(C), salt)$$

This ensures that the static noise associated with C is always the same independently of the query where C appears. The noise is "sticky" thereby preventing an attacker to send the same query multiple times, average out the results, and obtain a precise estimate of the private value (*averaging attack*) [18].

Dynamic noise. In the Diffix framework, every record in D is associated with a user ID, a unique string for that user.

These pseudonyms are used to compute the dynamic noise. Let $Q \equiv \mathsf{count}(C_1 \wedge \ldots \wedge C_h)$ be a query and C any condition C_i. The dynamic noise depends not only on C, but also on the *query set* of Q in the dataset D, i.e. the set of users which satisfies *all* conditions $C_1, \ldots, C_h$. More precisely, if the query set for Q on D is $S = \{uid_1, uid_2, \ldots, uid_m\}$, the dynamic noise for C ($\mathsf{dynamic}_Q[C]$) is generated from a normal distribution $\mathcal{N}(0,1)$ by the PRNG seeded with:

$$dynamic\_seed = \mathsf{XOR}(static\_seed_C, \mathsf{hash}(uid_1), \ldots, \mathsf{hash}(uid_m))$$

Note that we don't include D in the notation $\mathsf{dynamic}_Q[C]$, as the dataset is usually fixed and clear from the context.

The output $\widetilde{Q}(D)$ is therefore the realization of a random variable distributed as a normal distribution $\mathcal{N}(\mu, \sigma^2)$, with mean $\mu = Q(D)$ and variance $\sigma^2 = 2h$.

Example. Consider again the query

$$Q \equiv \mathsf{count}(gender = M \wedge age = 37 \wedge zip = 48828).$$

Diffix's output for Q on the database D is

$$\begin{aligned}\widetilde{Q}(D) = \quad & Q(D) \\ & + \mathsf{static}[gender = M] + \mathsf{dynamic}_Q[gender = M] \\ & + \mathsf{static}[age = 37] + \mathsf{dynamic}_Q[age = 37] \\ & + \mathsf{static}[zip = 48828] + \mathsf{dynamic}_Q[zip = 48828]\end{aligned}$$

where $\widetilde{Q}(D)$ is a random value drawn from a normal distribution $\mathcal{N}(Q(D), 6)$.

Static and dynamic noise layers are both needed to prevent *intersection attacks* [17, 18], a class of attacks that combine multiple queries to infer private attributes of records.

Bucket suppression. In addition to static and dynamic noise, Diffix implements another security measure called *bucket suppression*, similar to the classic query set size restriction. If the size of the query set is smaller than a certain threshold, the bucket suppression rejects the query. Previous research has shown that a fixed threshold constitutes a risk for privacy [22]. Diffix addresses this issue by using a *noisy* (and sticky to the query set) *threshold*. Specifically, suppose Diffix processes a query $Q \equiv \mathsf{count}(C_1 \wedge \ldots \wedge C_h)$. If $Q(D) \leq 2$, then the query gets suppressed. If $Q(D) > 1$, then Diffix computes a noisy threshold $T \sim \mathcal{N}(4, 1/2)$, using the seed:

$$threshold\_seed = \mathsf{XOR}(salt, \mathsf{hash}(uid_1), \ldots, \mathsf{hash}(uid_m))$$

If $Q(D) < T$, the query is suppressed; otherwise, the noisy output $\widetilde{Q}(D)$ is computed and sent to the analyst. In the original Diffix mechanism [17], the queries are said to be "silently suppressed" when censored by bucket suppression. This could mean that (1) a value of 0 is returned as result, (2) a random value is returned or (3) Diffix displays an error message. In this paper, we assume that a bucket-suppressed query will return a value of zero. This gives less information to a potential attacker than an error message, as a value of zero can be the result of either noise addition or bucket suppression. We consider that returning a random result affects utility too significantly to be applied in practice.

3 Noise-exploitation attacks

Our noise-exploitation attacks, which we call *differential* and *cloning*, are both based on three observations. First, since the noise is sticky, it is possible to *cancel out* part of it using multiple queries. Second, since the noise depends on the query set, the noise itself leaks information about the query set. Third, exploiting logical equivalence between some queries, it is possible to circumvent the "stickiness" of the noise by repeating (almost) the same query and consequently obtain independent noise samples. Our differential attack uses samples in a likelihood ratio test to discriminate between two probability distributions depending on the value of the private (also called secret) attribute. Our cloning attack relies on dummy conditions that conditionally strongly affect the output of the query depending on the value of the secret attribute.

3.1 Differential noise-exploitation attack

We first define further notations: with $A \subseteq \mathcal{A}_D$ a set of attributes, $x^{(A)}$ is the *restriction* of the record x to A, i.e. the vector one obtains after removing from x every entry for attributes that are not in A. For example, if $\mathcal{A}_D = \{gender, age, zip, HIV\}$, $x = (M, 27, 55416, 1)$ and $A = \{gender, age, HIV\}$, then $x^{(A)} = (M, 27, 1)$. If A contains a single attribute a, we simply write $x^{(a)}$. So, for example, $x^{(gender)} = M$.

For this attack, we make the following assumptions:

- H1 The attacker wants to find out some information about Bob, the victim. The attacker knows that Bob's record is in the dataset. We denote Bob's record by x. The attacker has access to the protected dataset only through Diffix.
- H2 The attacker knows all of Bob's attributes for some set of attributes A. Our *background knowledge* (also called auxiliary information in the literature) is the restricted record $x^{(A)}$. This is a standard assumption in the literature on re-identification attacks [23, 24].
- H3 The attacker wants to infer a secret attribute s about Bob, the victim. That is, she wants to infer the value of $x^{(s)}$, where $s \notin A$. For simplicity of notation, s is a binary attribute, i.e. $x^{(s)} \in \{0, 1\}$. This means that the attack can be seen as a classifier, with the output of the attack being negative if the algorithm returns $x^{(s)} = 0$ and positive if it returns $x^{(s)} = 1$.
  While we here focus on the binary case, our results fairly easily extend to non-binary cases.
- H4 There exists an oracle Unique that takes as input any restricted record $z^{(R)}$ and outputs whether $z^{(R)}$ is *unique*.

$\mathsf{Unique}(z^{(R)})$ = True if and only if there is no other record y in D such that $z^{(R)} = y^{(R)}$.

For the attack to succeed, we first need to ensure that the background knowledge $x^{(A)}$ uniquely identifies Bob. This is given to us by the oracle $\mathsf{Unique}(x^{(A)})$. In this attack, we only attempt to infer secret attributes of records that are unique in the dataset. The cloning attack, presented later, extends this by requiring a weaker notion of uniqueness, which we call value-uniqueness.

While the existence of such an oracle is a strong assumption, it is weaker than Diffix's claims to protect against an "analyst [that] has full access to the inference dataset" [18], where the *inference dataset* is the original dataset with only the secret attribute $x^{(s)}$ removed. If the attacker has access to every record, she can easily verify that no other record shares the same restricted record $x^{(A)}$.

Firstly, the attack needs to bypass bucket suppression. For example, an attacker could ask how many records have both the background knowledge $x^{(A)}$ and the private attribute $s = 0$:

$$Q \equiv \mathsf{count}(a_1 = x_1 \wedge \ldots \wedge a_k = x_k \wedge s = 0). \tag{2}$$

with $A = \{a_1, \ldots, a_k\}$ and $x^{(A)} = (x_1, \ldots, x_k)$.

While an accurate answer to Q would immediately disclose the value of $x^{(s)}$, since $Q(D)$ can be either 0 (if $x^{(s)} = 1$) or 1 (if $x^{(s)} = 0$), this query will always be blocked by bucket suppression since the query set is either empty or {Bob}.

Intersection attacks have been proposed in the literature to circumvent similar kinds of restrictions [25]. Picking an attribute, e.g. a_1, we can define the following queries:

$$Q_1 \equiv \mathsf{count}(a_2 = x_2 \wedge \ldots \wedge a_k = x_k \wedge s = 0) \tag{3}$$

$$Q_1' \equiv \mathsf{count}(a_1 \neq x_1 \wedge a_2 = x_2 \wedge \ldots \wedge \ldots \wedge a_k = x_k \wedge s = 0) \tag{4}$$

As the record x is unique, by assumption, it is the only record that can differ between Q_1 and Q_1'. This allows us to directly compute $Q(D)$:

$$Q(D) = Q_1(D) - Q_1'(D) = \begin{cases} 0 & \text{if } x^{(s)} = 1 \\ 1 & \text{if } x^{(s)} = 0 \end{cases} \tag{5}$$

To prevent this vulnerability[1], Diffix adds static and dynamic noises:

$$\begin{aligned} \widetilde{Q}_1(D) = \quad & Q_1(D) \\ & + \sum_{i=2}^{k} \mathsf{static}[a_i = x_i] + \mathsf{static}[s = 0] \\ & + \sum_{i=2}^{k} \mathsf{dynamic}_{Q_1}[a_i = x_i] + \mathsf{dynamic}_{Q_1}[s = 0] \end{aligned}$$

and

$$\begin{aligned} \widetilde{Q}_1'(D) = \quad & Q_1'(D) \\ & + \mathsf{static}[a_1 \neq x_1] + \mathsf{dynamic}_{Q_1'}[a_1 \neq x_1] \\ & + \sum_{i=2}^{k} \mathsf{static}[a_i = x_i] + \mathsf{static}[s = 0] \\ & + \sum_{i=2}^{k} \mathsf{dynamic}_{Q_1'}[a_i = x_i] + \mathsf{dynamic}_{Q_1'}[s = 0]. \end{aligned}$$

The first part of our attack relies on noticing that $k-1$ static noise layers cancel out. Let

$$q_1 = \widetilde{Q}_1(D) - \widetilde{Q}_1'(D).$$

Then:

$$\begin{aligned} q_1 = \quad & Q_1(D) - Q_1'(D) \\ & - \mathsf{static}[a_1 \neq x_1] - \mathsf{dynamic}_{Q_1'}[a_1 \neq x_1] \qquad (\textit{fixed}) \\ & + \sum_{i=2}^{k} \mathsf{dynamic}_{Q_1}[a_i = x_i] + \mathsf{dynamic}_{Q_1}[s = 0] \\ & \qquad\qquad (\textit{dynamic}_{Q_1}) \\ & - \sum_{i=2}^{k} \mathsf{dynamic}_{Q_1'}[a_i = x_i] - \mathsf{dynamic}_{Q_1'}[s = 0] \\ & \qquad\qquad (\textit{dynamic}_{Q_1'}) \end{aligned}$$

leaving us with $2k+2$ noise layers: $\textit{fixed} \sim \mathcal{N}(0,2)$, $\textit{dynamic}_{Q_1} \sim \mathcal{N}(0,k)$, and $\textit{dynamic}_{Q_1'} \sim \mathcal{N}(0,k)$.

The second part of the attack relies on the fact that both $\mathsf{dynamic}_{Q_1}[a_i = x_i]$ and $\mathsf{dynamic}_{Q_1'}[a_i = x_i]$ (resp. $\mathsf{dynamic}_{Q_1}[s = 0]$ and $\mathsf{dynamic}_{Q_1'}[s = 0]$) relate to the same condition "$a_i = x_i$" (resp. "$s = 0$"). This means that the noise values for Q_1 and Q_1' will cancel out if Q_1 and Q_1' have the same query set. Therefore either $Q_1(D) - Q_1'(D) = 0$, and the $2k$ dynamic noise layers cancel out, or $Q_1(D) - Q_1'(D) = 1$, and no dynamic noise layer is canceled out:

$$q_1 \sim \begin{cases} \mathcal{N}(0,2) & \text{if } x^{(s)} = 1 \\ \mathcal{N}(1,2k+2) & \text{if } x^{(s)} = 0 \end{cases} \tag{6}$$

Using this result, an attacker can run a likelihood ratio test (see Appendix A) to estimate whether q_1 is distributed as $\mathcal{N}(0,2)$ or $\mathcal{N}(1,2k+2)$ and predict the value of $x^{(s)}$. The larger k is, the easier it becomes to discriminate between the two distributions. This alone already allows the attacker to infer Bob's secret, $x^{(s)}$, with better than random accuracy.

The third part of the attack allows us to strongly improve the accuracy of our inference. While the stickiness of the noise prevents us from running the same query again to collect more sample, we circumvent it by using different pairs of queries for which equation (6) is still true. Specifically, instead of removing (resp. negating) the condition $a_1 = x_1$, we remove

[1] For the intersection attack to be successful, both Q_1 and Q_1' need to be large enough to not trigger the bucket suppression. We discuss this assumption later on.

(resp. negate) other conditions $a_j = x_j$ for $j \leq k$, obtaining queries Q_j (resp. Q'_j) for $j \leq k$. In our notation:

$$Q_j \equiv \mathsf{count}\left(\bigwedge_{\substack{i=1\\ i\neq j}}^{k} a_i = x_i \wedge s = 0\right) \tag{7}$$

$$Q'_j \equiv \mathsf{count}\left(\bigwedge_{\substack{i=1\\ i\neq j}}^{k} a_i = x_i \wedge a_j \neq x_j \wedge s = 0\right) \tag{8}$$

Running all queries $\{(Q_j, Q'_j)\}_{j\leq k}$, the attacker collects a vector of realizations $\{q_j\}_{j\leq k}$ where $q_j = \widetilde{Q}_j(D) - \widetilde{Q}'_j(D)$. All q_j values are computed from different queries, which generate different noises. Hence, the noises all have different values (with probability 1). Nevertheless, the equation (6) is still true for each q_j, so we can combine them as k different samples from the same distribution, and estimate the value of $Q(D)$.

Finally, replacing the "$s = 0$" condition with "$s = 1$" in every pair (Q_j, Q'_j) defines k new pairs of queries $\{(R_j, R'_j)\}_{j\leq k}$. This allows us to obtain k more samples $\{r_j\}_{j\leq k}$ (with different noises and inverted results in equation (6)). This gives us a total of $2k$ samples before bucket suppression (see Appendix A), generated by issuing $4k$ queries.

On a technical note, observe that in principle we cannot be certain that the q_j's (resp. r_j's) are all independent samples, because two different queries Q_j and Q_l (resp. R_j and R_l) with $j \neq l$ might have the same query set. In that case, the dynamic noise layers associated to the same conditions would have the same values, and hence the total dynamic noises of the two queries would be heavily correlated. While this affects the accuracy of the likelihood ratio test, the impact is negligible in practice. Hereafter, we always assume that the samples are independent.

Full differential attack. For larger values of k, the queries used by the differential attack contain many conditions, and hence potentially select a low number of records. Dependending on the dataset, this might result in a large fraction of queries being bucket suppressed, leaving the attacker with few or no samples for the likelihood ratio test. To counteract this effect, we integrate the attack with a subset exploration step to obtain a *full differential attack*. Assume that the attacker knows a set A^* of the correct attributes for the victim with $|A^*| = k^*$, i.e. the background knowledge is $x^{(A^*)}$. The full attack proceeds as follows. The algorithm selects random subsets of A^* until it finds a subset $A \subseteq A^*$ such that $\mathsf{Unique}(x^{(A)})$ returns True. It then performs the differential attack using $x^{(A)}$ as background knowledge. For the likelihood ratio test, the attack considers only query pairs $(\widetilde{Q}_j(D), \widetilde{Q}'_j(D))$ or $(\widetilde{R}_j(D), \widetilde{R}'_j(D))$ which have outputs larger than zero in both entries. If no such pair exists, the algorithm samples a new subset A and iterates the procedure. If no feasible subset is found, the algorithm outputs NonAttackable. The subsets of A^* are sampled by decreasing size, as bucket suppression is less likely for lower values of k (but on the other hand the likelihood ratio test is less accurate).

The procedure FullDifferentialAttack presents in detail the algorithm outlined above.

Procedure DifferentialAttack($A, x^{(A)}, s$)

Input: known attributes (names A and values $x^{(A)}$), secret s

Output: True if $x^{(s)} = 1$, False if $x^{(s)} = 0$, NoSamples if $x^{(A)}$ does not yield any positive sample

$k \leftarrow |A|$, $\mathcal{Q} \leftarrow \emptyset$, $\mathcal{R} \leftarrow \emptyset$
for $j \leftarrow 1$ **to** k **do**
  $I \leftarrow \{i \in [1,k] \mid i \neq j\}$
  $\widetilde{Q} \leftarrow \mathsf{count}\left(\bigwedge_{i\in I} a_i = x_i \wedge s = 0\right)$
  $\widetilde{Q}' \leftarrow \mathsf{count}\left(\bigwedge_{i\in I} a_i = x_i \wedge a_j \neq x_j \wedge s = 0\right)$
  if $\widetilde{Q} > 0$ and $\widetilde{Q}' > 0$ **then**
    $q_j \leftarrow \widetilde{Q} - \widetilde{Q}'$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{q_j\}$
  end if
end for
for $j \leftarrow 1$ **to** k **do**
  $I \leftarrow \{i \in [1,k] \mid i \neq j\}$
  $\widetilde{R} \leftarrow \mathsf{count}\left(\bigwedge_{i\in I} a_i = x_i \wedge s = 1\right)$
  $\widetilde{R}' \leftarrow \mathsf{count}\left(\bigwedge_{i\in I} a_i = x_i \wedge a_j \neq x_j \wedge s = 1\right)$
  if $\widetilde{R} > 0$ and $\widetilde{R}' > 0$ **then**
    $r_j \leftarrow \widetilde{R} - \widetilde{R}'$
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{r_j\}$
  end if
end for
if $\mathcal{Q} = \emptyset$ and $\mathcal{R} = \emptyset$ **then**
  return NoSamples
end if
$f \leftarrow$ PDF of $\mathcal{N}(0,2)$, $\quad g \leftarrow$ PDF of $\mathcal{N}(1, 2k+2)$
$L \leftarrow \prod_{q\in\mathcal{Q}} \frac{f(q)}{g(q)} \prod_{r\in\mathcal{R}} \frac{g(r)}{f(r)}$
return $L \geq 1$

3.2 Cloning noise-exploitation attack

In this section, we present an extension of the differential noise-exploitation attack, which we call *cloning attack*. This attack adds dummy conditions, that don't affect the query set, to queries, in such a way that several queries with different dummy conditions will have either identical or very different results conditional to the secret attribute's value.

We first introduce some new notations and definitions. Denoting by x the victim's entire record, the attacker's background information is now $x^{(A)} = (x^{(A')}, x^{(u)})$ with $A = A' \cup \{u\}$ and $|A| = k$. We use the shorthand (A', u) for $A' \cup \{u\}$. We also define a restricted record $z^{(A)}$ to be *value-unique* for the attribute s in D if $y^{(A)} = z^{(A)}$ implies $y^{(s)} = z^{(s)}$. That is,

Procedure FullDifferentialAttack($A^*, x^{(A^*)}, s$)

Input: known attributes (names A^* and values $x^{(A^*)}$), secret s
Output: True if $x^{(s)} = 1$, False if $x^{(s)} = 0$, NonAttackable if $x^{(A^*)}$ is not attackable

```
for k ← |A*| to 1 do
    for iter ← 1 to 100 do
        A ← RandomSubsetOfSize(A*, k)
        if Unique(x^(A)) and
          DifferentialAttack(A, x^(A), s) ≠ NoSamples
          then
            return DifferentialAttack(A, x^(A), s)
        end if
    end for
end for
return NonAttackable
```

every record that shares the same attributes of the restricted record $z^{(A)}$ holds the same value for the secret attribute s. To simplify the notation, we write $A' = x^{(A')}$ to indicate the condition that all attributes in A' must match the ones in $x^{(A')}$, i.e. $\wedge_{a \in A'} a = x^{(a)}$

The attack addresses several limitations of the differential attack, making it much stronger in practice.

First, the cloning attack does not require an oracle to confirm that the background information uniquely identifies a user. Instead, it replaces the oracle with a heuristic to automatically validate the assumptions.

Second, the differential attack has to ignore any pair $(\widetilde{Q}_j(D), \widetilde{Q}'_j(D))$ where at least one of the entries is not positive, as it cannot tell whether the null output comes from noise addition or from bucket suppression. This significantly reduces the total number of samples used. Our cloning attack instead uses "dummy conditions" that do not impact the user set. As queries now only differ in the dummy conditions, the corresponding query sets will always be identical. This allows us to rule out bucket suppression in case at least one output is greater than zero.

Third, while the differential attack requires records to be unique, the cloning attack only requires records to be value-unique. This is a weaker condition and makes the cloning attacks effective on a larger set of users.

Fourth, the cloning attack only requires that the set of users who share all attributes in $x^{(A')}$ is "large enough" to not be bucket suppressed. This is a weaker assumption than for the differential attack, where this needed to hold for a large number of subsets of A with one attribute removed. Furthermore, the cloning attack validates this automatically (and thus prevents bucket suppression) with high confidence.

While much stronger, the attack relies on the attacker being able to produce a set of distinct *dummy conditions* $\Delta = \{\Delta_j\}_{1 \le j \le |\Delta|}$, where each Δ_j is an SQL statement such that the set of users matching $A' = x^{(A')}$ is the same as the set of users matching $A' = x^{(A')} \wedge \Delta_j$. In section 5, we discuss how dummy conditions are easy to obtain, slow to detect, and how automatically filtering them might introduce new vulnerabilities.

Description of the attack. For each dummy condition Δ_j, we define the two queries:

$$Q_j \equiv \mathsf{count}\left(A' = x^{(A')} \wedge \Delta_j \wedge s = 0\right) \tag{9}$$

$$Q'_j \equiv \mathsf{count}\left(A' = x^{(A')} \wedge \Delta_j \wedge u \neq x^{(u)} \wedge s = 0\right) \tag{10}$$

With $q_j = \widetilde{Q}_j(D) - \widetilde{Q}'_j(D)$, we have:

$$\begin{aligned} q_j = \quad & Q_j(D) - Q'_j(D) \\ & - \mathsf{static}[u \neq x^{(u)}] \; - \mathsf{dynamic}_{Q_j}[u \neq x^{(u)}] \\ & + \sum_{i \in A'} \mathsf{dynamic}_{Q_j}[a^{(i)} = x^{(i)}] + \mathsf{dynamic}_{Q_j}[s = 0] \\ & - \sum_{i \in A'} \mathsf{dynamic}_{Q'_j}[a^{(i)} = x^{(i)}] - \mathsf{dynamic}_{Q'_j}[s = 0] \\ & + \left(\mathsf{dynamic}_{Q_j}[\Delta_j] - \mathsf{dynamic}_{Q'_j}[\Delta_j]\right) \end{aligned}$$

By the same argument we presented for the differential attack, if $x^{(s)} = 1$ then $Q_j(D) = Q'_j(D)$ and most dynamic and static noises cancel out, giving:

$$q_j = -\mathsf{static}[u \neq x^{(u)}] - \mathsf{dynamic}_{Q_j}[u \neq x^{(u)}] \tag{11}$$

As this value does not depend on the dummy condition used, we have that $q_1 = q_2 = \cdots = q_{|\Delta|}$.

On the contrary, if $x^{(s)} = 0$, then the noise layers do not cancel out with probability 1. As the noise values given by $\mathsf{dynamic}_{Q_j}[\Delta_j]$ and $\mathsf{dynamic}_{Q'_j}[\Delta_j]$ depend on Δ_j, the probability that all (or any) q_j are equal is zero.

We can therefore complete the attack by inferring that $x^{(s)} = 1$ if $q_1 = \cdots = q_{|\Delta|}$, and $x^{(s)} = 0$ otherwise. Under the current assumptions, the attack always infers the correct value with 100% confidence (up to pseudo-random collisions in Diffix's noise addition mechanism).

Robustness against rounding. In the previous section, we follow the Diffix papers [17, 18] and assume that $\widetilde{Q}(D)$ is returned directly without any rounding, admitting also negative values. We now consider the case where results are rounded to the nearest nonnegative integer, and propose a simple modification of our attack that accounts for this.

When the results of the queries Q_j and Q'_j are rounded, the corresponding q_j might not be identical if $x^{(s)} = 0$. However, the q_js will vary less if $x^{(s)} = 1$ than if $x^{(s)} = 0$. Hence, instead of checking if $q_1 = \ldots = q_{|\Delta|}$, we check if the q_j values are "similar" to one another. While for high values of k this is easy to detect, the total variance of the noise for low values

of k is small, making it harder to distinguish between the two hypotheses (i.e. whether the q_j values are "similar" or not). To overcome this issue, we "amplify" the noise for each query: instead of adding a single dummy condition Δ_j to the queries for Q_j and Q'_j, we add the conjunction $\wedge_{l\neq j}\Delta_l$:

$$Q_j \equiv \mathsf{count}\left(A' = x^{(A')} \wedge \bigwedge_{l\neq j} \Delta_l \wedge s = 0\right) \tag{12}$$

$$Q'_j \equiv \mathsf{count}\left(A' = x^{(A')} \wedge \bigwedge_{l\neq j} \Delta_l \wedge u \neq x^{(u)} \wedge s = 0\right) \tag{13}$$

This increases the total variance of the noise in q_j in the $x^{(s)} = 0$ case, making it easy to distinguish between the two hypotheses: all the q_j values will be very similar if $x^{(s)} = 1$ and fluctuate heavily if $x^{(s)} = 0$. Measuring the sample variance S^2 of $\{q_j\}_{1\leq j\leq|\Delta|}$, we infer that $x^{(s)} = 1$ if $S^2 \leq \sigma^*$, and $x^{(s)} = 0$ otherwise with a cutoff threshold σ^* chosen by the attacker. We include an empirical analysis of σ^* in the full version of this manuscript. The cloning attack is described in detail in the procedure CloningAttack.

Procedure CloningAttack($A', u, x^{(A',u)}, \Delta, s, v$)

Input: known attributes (names A', u and values $x^{(A',u)}$), dummy conditions Δ, secret s and target value v
Output: True if $x^{(s)} = v$, False if $x^{(s)} \neq v$

for $j \leftarrow 1$ **to** $|\Delta|$ **do**
  $\varphi \leftarrow A' = x^{(A')} \wedge \bigwedge_{l\neq j}\Delta_l$
  $\widetilde{Q} \leftarrow \mathsf{count}(\varphi \wedge s \neq v)$
  $\widetilde{Q}' \leftarrow \mathsf{count}\left(\varphi \wedge u \neq x^{(u)} \wedge s \neq v\right)$
  $q_j \leftarrow \widetilde{Q} - \widetilde{Q}'$
end for
$\overline{r} \leftarrow \frac{1}{|\Delta|}\sum_{j=1}^{|\Delta|} q_j, \quad S^2 \leftarrow \frac{1}{|\Delta|-1}\sum_{j=1}^{|\Delta|}(q_j - \overline{r})^2$
return $S^2 \leq \sigma^*$

Automated validation of the assumption. The cloning attack relies on two assumptions on the attacker's background knowledge $x^{(A',u)}$:

1. The queries $\{Q_j\}_{1\leq j\leq|\Delta|}$ and $\{Q'_j\}_{1\leq j\leq|\Delta|}$ in equations (12) and (13) are not bucket suppressed.
2. The user is value-unique in the dataset according to (A', u) for the secret attribute s.

We here propose procedures for an attacker to determine whether (A', u) satisfies the two assumptions with high probability.

Validating the first assumption can be done easily by submitting queries $\{Q_j\}_{1\leq j\leq|\Delta|}$ and $\{Q'_j\}_{1\leq j\leq|\Delta|}$ to Diffix. Recall that the threshold for bucket suppression for a query depends only on the corresponding query set. All the queries in $\{Q_j\}_{1\leq j\leq|\Delta|}$ have the same query set, and the same applies for $\{Q'_j\}_{1\leq j\leq|\Delta|}$. Hence, if *any* query Q_j is bucket suppressed (i.e. has output zero), then *all* queries in $\{Q_j\}_{1\leq j\leq|\Delta|}$ must have output zero, and similarly for $\{Q'_j\}_{1\leq j\leq|\Delta|}$. Thus, if *any* query Q_j and *any* query Q'_j have output higher than zero, we are sure that no query was bucket suppressed, and hence all q_j's are valid samples. The test is considered passed in this case, and failed otherwise. See the algorithm NoBucketSuppression for an implementation example.

Validating the second assumption relies on a heuristic. We run the query:

$$\mathsf{count}(A' = x^{(A')} \wedge u = x^{(u)})$$

and consider the assumption validated if the output is zero, and not otherwise. The idea is that if the output is larger than zero, then the query was not bucket suppressed, and many users are likely share the same attributes $x^{(A',u)}$, meaning that $x^{(A',u)}$ is unlikely to be value-unique. Experiments in section 4 show that this heuristic works very well on real-world datasets.

Procedure NoBucketSuppression($A', u, x^{(A',u)}, \Delta, s, v$)

Input: known attributes (names A', u and values $x^{(A',u)}$), dummy conditions Δ, secret s and target value v
Output: True if (A', u) passes the tests and is deemed to satisfy assumption 1, False otherwise

$ok_Q \leftarrow 0, ok_{Q'} \leftarrow 0$
for $j \leftarrow 1$ **to** $|\Delta|$ **do**
  $\varphi \leftarrow A' = x^{(A')} \wedge \bigwedge_{l\neq j}\Delta_l$
  $\widetilde{Q} \leftarrow \mathsf{count}(\varphi \wedge s \neq v)$
  $\widetilde{Q}' \leftarrow \mathsf{count}\left(\varphi \wedge u \neq x^{(u)} \wedge s \neq v\right)$
  if $\widetilde{Q} > 0$ **then**
    $ok_Q \leftarrow 1$
  end if
  if $\widetilde{Q}' > 0$ **then**
    $ok_{Q'} \leftarrow 1$
  end if
end for
return $ok_Q = 1$ & $ok_{Q'} = 1$

Procedure ValueUnique($A', u, x^{(A',u)}$)

Input: known attributes (names A', u and values $x^{(A',u)}$)
Output: True if (A', u) passes the tests and is deemed to satisfy assumption 2, False otherwise

$\widetilde{Q} \leftarrow \mathsf{count}(A' = x^{(A')} \wedge u = x^{(u)})$
return $\widetilde{Q} = 0$

The procedures CloningAttack and NoBucketSuppression both issue (the same) $2|\Delta|$ queries, while ValueUnique uses

only one query. Validating the assumptions and performing the attack thus requires only $2|\Delta|+1$ queries, for a given set of attributes (A', u). We empirically obtain accuracy above 93.3% with $|\Delta|$ as low as 10 (see section 4 and Appendix B).

Full cloning attack. Combining procedures CloningAttack, NoBucketSuppression and ValueUnique, we design a fully fledged procedure FullCloningAttack that performs the entire attack under the following assumptions:

- H1 The attacker knows that the victim's record x is in the dataset.
- H2 The attacker knows a set A^* of the correct attributes for the victim with $|A^*| = k^*$, i.e. the background knowledge is $x^{(A^*)}$.
- H3 The secret attribute $x^{(s)}$ is a binary attribute.

The full cloning attack includes a subset exploration step similar to the one used in the full differential attack. The algorithm selects random subsets A' of A^* (and an element u from $A^* \setminus A'$ at random) by decreasing size until it finds a subset that passes both tests, upon which it then performs the attack using $x^{(A',u)}$ as background knowledge. If no feasible subset is found, the algorithm outputs NonAttackable.

Procedure FullCloningAttack$(A^*, x^{(A^*)}, \Delta, s, v)$

Input: known attributes (names A^* and values $x^{(A^*)}$), dummy conditions Δ, secret s and target value v
Output: True if $x^{(s)} = v$, False if $x^{(s)} \neq v$

```
for k ← |A*| to 1 do
    for iter ← 1 to 100 do
        A' ← RandomSubsetOfSize(A*, k − 1)
        u ← RandomElement(A* \ A')
        if NoBucketSuppression(A', u, x^(A), Δ, s, v) and
          ValueUnique(A', u, x^(A*)) then
            return CloningAttack(A', u, x^(A), Δ, s, v)
        end if
    end for
end for
return NonAttackable
```

Reducing the number of queries. While Diffix allows each analyst to send arbitrarily many queries, we study how many queries are required to perform the cloning attack in practice. In Appendix B we present a heuristic that reduces the median number of queries by a factor of 100. Using this heuristic, the attack targets 55.4% of the users in the dataset, achieving 91.7% accuracy with a maximum of 32 queries per user in our experiments.

4 Experiments

In order to assess the effectiveness of our attacks, we implemented Diffix's mechanism for counting queries as described in the original paper [18]. The implementation outputs zero when queries are bucket-suppressed and results are rounded to the nearest nonnegative integer. We apply our attacks to four datasets and an additional synthetic dataset on which the assumptions of the differential attack are always validated.

4.1 Description of the datasets

In our experiments, we use the following datasets:

1. ADULT: U.S. Census dataset with 30,162 records and 11 attributes, incl. salary class as secret attribute [26].
2. CREDIT: credit card application dataset with 690 records and 16 attributes, incl. accepted credit as secret attribute [27].
3. CENSUS: U.S. Census dataset with 199,523 records and 42 attributes, incl. total personal income (digitized, null income as negative condition) as secret attribute [28].
4. CDR: synthetic collection of phone metadata with 2,000,000 records generated using real-world data for human behaviour and the geography of the UK for the location of antennas. Every user is a record of 11,674,870 binary attributes (an attribute being whether a user was geographically present at a certain place and time, and placed a call or received a text message). As the vast majority of the attributes in a record are null, the distribution of values for a random attribute is heavily skewed towards zero. To obtain a balanced experiment, for 50% of runs we select as the secret attribute a pair $(location, time)$ where the user was present, and for the other 50% we select a pair where the user was absent.

4.2 Differential noise-exploitation attack

Evaluation of the attack alone. We first test the differential attack on a synthetic dataset where all users satisfy the uniqueness assumption.

In the $\mathsf{Complete}_k$ dataset, every user is unique according to k attributes (excluding the secret attribute), whereas $k-1$ attributes always identify a *larger* set of users. This ensures that (*i*) every user is vulnerable to the attack and (*ii*) bucket suppression is unlikely to be triggered by the attack queries. To create the dataset, we fix an integer B and generate every possible $k-$tuple whose values are in $\{1,\ldots,B\}$. We then append to each tuple a random value of either 0 or 1 for the secret attribute. $\mathsf{Complete}_k$ contains B^k records, one for each combination of k attributes. For our experiments, we set $B = 12$, to ensure that close to no bucket suppression occurs. For computational reasons, as the size of the dataset in memory grows as B^k, the maximum k we can use is limited to 6.

Fig. 2 compares the accuracy $\mathrm{acc}(k)$ of the attack, knowing k attributes, on $\mathsf{Complete}_k$ with the theoretical accuracy. The procedure we use here is DifferentialAttack, which does *not* include subset exploration and has no access to the oracle.

Hence, this experiment simulates a realistic attacker. We report the empirical fraction of users whose secret attribute is correctly predicted, estimated by performing the differential attack on a sample of 1000 users. We also report the theoretical distribution of accuracy, (*i*) without rounding (closed-form expression, see Appendix A) and (*ii*) with rounding (numerical simulation, see Appendix A). For the Complete$_k$ dataset, the accuracy reaches 92.6% for 5 points. Even knowing only $k = 2$ attributes, the accuracy is above 66% both theoretically and empirically. While rounding has close to no effect on the theoretical accuracy of the attack, comparing the Complete$_k$ with the Theoretical (rounding) curves shows that bucket suppression and potential correlations between the samples in empirical experiments noticeably decrease the accuracy.

Evaluation on real-world datasets. We now evaluate the accuracy of the differential attack on 1,000 users selected at random in each of the four datasets. Contrary to the synthetic experiment, bucket suppression is more prevalent on real-world datasets. Therefore, we run the FullDifferentialAttack algorithm, knowing k^* attributes A^* that are selected at random for each record. If the attack outputs NonAttackable, the secret attribute is predicted at random (uniformly).

Fig. 3 shows, for each dataset and knowing k^* attributes, the percentage of unique individuals and the percentage of individuals, in each dataset, for which the secret is correctly inferred. The latter divided by the the former gives us the accuracy of our attack. Our attack realizes an accuracy of 68.4% for ADULT with $k^* = 10$, 64.0% for CREDIT with $k^* = 15$, 68.8% for CENSUS with $k^* = 40$, and 68.8% for CDR with $k^* = 6$.

Observe that the fraction of correctly inferred attributes

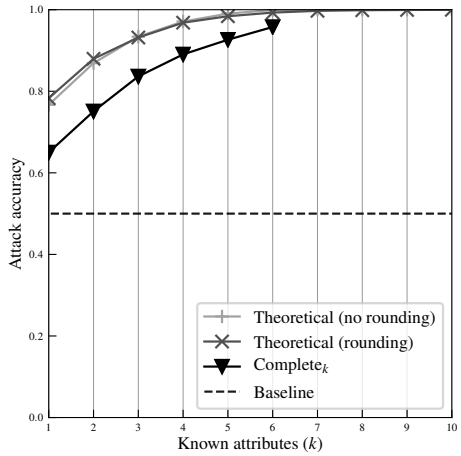

Figure 2: Accuracy of the differential attack when the uniqueness assumption is always validated and with balanced truth values. The baseline accuracy is 0.5 and represents the expected success rate when randomly predicting the secret attribute using a uniform prior.

Dataset	Attributes (k^*)	Value-unique	Predicted attackable	accuracy$_{pa}$	accuracy$_{all}$
ADULT	10	93.0%	96.8%	93.3%	87.0%
CREDIT	15	100.0%	100.0%	97.0%	97.0%
CENSUS	40	99.7%	94.6%	97.1%	91.6%
CDR	6	100.0%	100.0%	91.3%	91.3%

Table 1: Empirical results of the cloning attack on four real-world datasets.

plateaus with larger k^* for the CREDIT, CENSUS and CDR datasets. The reason is that, on these datasets, most users are unique for larger values of k^*. As explained in section 3, this makes bucket suppression more prevalent and reduces the total number of samples for the likelihood ratio test, which is a limitation of the differential attack.

4.3 Cloning noise-exploitation attack

We implement the attack as described by algorithm FullCloningAttack. As before, for each value of k^* we select 1,000 users at random, and for each user a random subset A^* of their attributes of size k^*. A^* represents the total number of attributes known to the attacker about the victim.

We set a threshold $\sigma^* = 0.7$ for the variance cutoff (see full version). We generate $|\Delta| = 10$ dummy conditions for $x_1 = a_1$ of the form $x_1 \neq b_j$ for $j \leq |\Delta|$, with b_j being some plausible values for x_1 different from a_1. We present the results when the attacker knows enough attributes (k^*) to identify every user, or up to all available attributes in the dataset.

Table 1 shows the proportion of records that are value-unique (third column) and fraction of the value-unique records that are predicted as attackable by procedures NoBucketSuppression and ValueUnique (fourth column). We then perform the cloning attack on all records that are predicted as attackable and report accuracy$_{pa}$, the fraction of predicted attackable records whose secret attribute was successfully inferred (fifth column). For completeness, we also report accuracy$_{all}$, the fraction of all records in the datasets (including the ones deemed NonAttackable) whose secret attribute was successfully inferred (last column).

Table 1 shows that the cloning attack—including the assumption validation step—performs really well on all datasets considered, between 87.0 and 97.0% of secret attributes and 97.0% on the CREDIT dataset when knowing 15 attributes.

Fig. 4 shows, knowing k^* attributes, the fraction of all records that are value-unique, predicted attackable, and correctly inferred. The curves for value-unique users and for predicted attackable users are always very close, suggesting

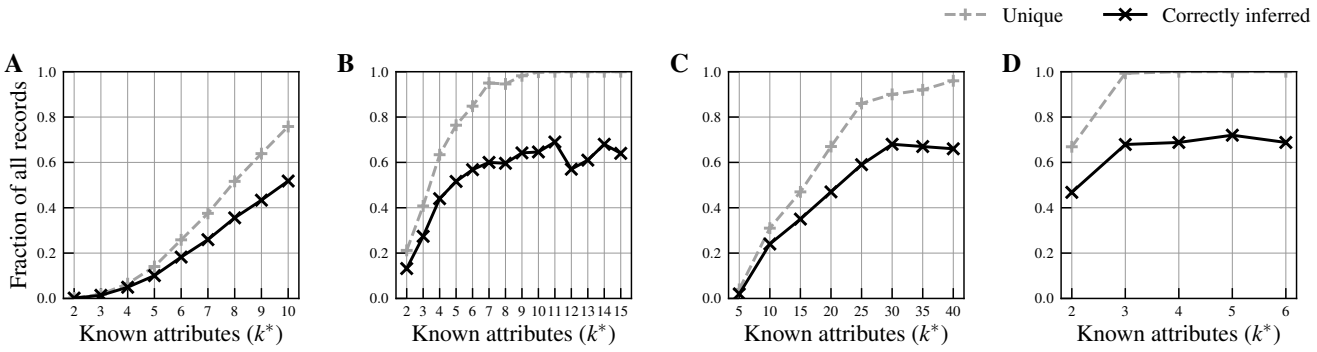

Figure 3: Results of the differential attack for the (A) ADULT, (B) CREDIT, (C) CENSUS, and (D) CDR datasets.

that the assumption validation step is effective. Out of all records predicted as attackable, most of them are correctly inferred, demonstrating that the attack works on targeted records across all k. For the CREDIT dataset, with only six attributes, the attack reaches the inference step for 95% of the users in the dataset, and correctly infers the secret attribute for 93% of the total records.

5 Discussion

5.1 Value-uniqueness and attribute predictability

Value-uniqueness plays an important role in the cloning attack. As Fig. 4 shows, it is a valid assumption for real-world datasets.

Value-uniqueness means that a group of people who share the same attributes also share the same secret attribute. If this group were to be large enough, the noise added by Diffix might not be enough to hide the secret attribute, which could then be revealed by using a simple count query. While this might be true for some datasets, it is not the case for any of the datasets we considered. For instance, the average size of the value-unique class (i.e. the set of value-unique users sharing the same restricted record) in the ADULT dataset is 1.44, with no class containing more than 4 users and similar numbers for the other datasets. This means that, most of the times, value-unique users are simply *unique*.

If secret attributes are predictable from the other attributes, a trained machine learning classifier could predict them with potentially high accuracy[2]. Despite our datasets coming from the machine learning literature, our attack does not rely at all on the predictability of secret attributes and performs equally well if no correlation at all exists between attributes and the secret attribute. We run our attack on a modified version of the ADULT dataset where sensitive attributes have been randomly sampled, thereby theoretically destroying any correlation. Our attacks perform as well on this modified dataset as on the original dataset. These results are included in the full version of this manuscript.

[2]Whether this would constitute a privacy attack is debated [29].

5.2 Producing and detecting dummy conditions

The cloning attack requires the attacker to provide a set of dummy conditions that affect the noise addition without affecting the query set. These conditions can be syntactic (e.g., $age \geq 15$ for the query $age = 23$), semantic (e.g., $status \neq retired$ for $age = 23$), or pragmatic (e.g., $age \neq 15$ against a database containing only adult individuals).

When the language is rich enough, detecting redundant clauses is not a trivial task. At the same time, the richer the syntax is, the more utility an analyst gets out of the system. Diffix offers a fairly rich syntax including boolean expression, GROUP BY, JOIN, seven aggregation functions, set membership, fourteen string functions, and ten maths functions. In this context, automatically detecting dummy conditions would likely require iterating recursively through every condition and evaluating the query with and without them, a costly operation. Moreover, if dummy conditions are detected by evaluating them on the dataset, filtering them might not be safe. Removing semantic and pragmatic dummy conditions from a query would indeed reduce the total variance of the added noise and leak information about the dataset itself (e.g., only adult individuals are present if the condition $age \neq 15$ is always removed).

5.3 Improving the attacks

The cloning attack can be modified to run a double NoBucketSuppression test: once with $v = 0$ and once with $v = 1$. If both pass and the ValueUnique test is passed as well, the attack proceeds with the actual inference procedure CloningAttack for both $v = 0$ and $v = 1$. The attack then makes a guess only if both inferences return the same value, and continues with the subset exploration otherwise (deeming the user NonAttackable if no working set of attributes is found).

We found that this modified attack improves the accuracy$_{\text{pa}}$ figure on all datasets (e.g. from 93.3% to 97.3% for ADULT,

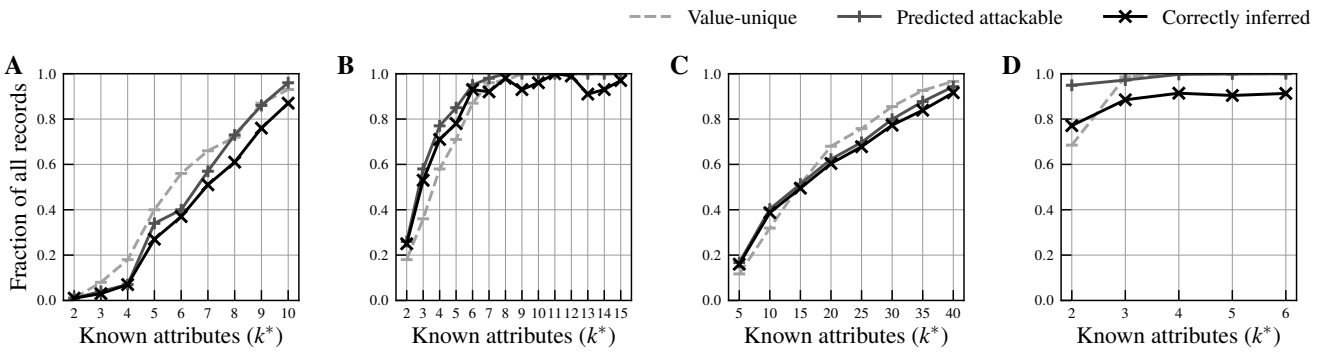

Figure 4: Results of the cloning attack for the (A) ADULT, (B) CREDIT, (C) CENSUS, and (D) CDR datasets.

all results available in the full version). However, double tests are more likely to fail, meaning that less users are predicted as attackable (from 96.8% to 87% in ADULT). Because of bucket suppression, this effect is particularly strong for datasets where the overall distribution of the secret attribute is very skewed, such as the CDR dataset where the number of predicted attackable users goes from 100% to 8.5%. Depending on the aims of the attacker (precision versus coverage), she might prefer the original or the double version of the cloning attack.

Other improvements. To properly quantify the strength of our attacks, none of them use prior knowledge on the distribution of the secret attribute. In practice, an attacker might want to use this information, e.g. obtaining it by querying Diffix. We discuss this in the full version of the manuscript. We also discuss how to generate more samples for the differential attack and outline how to generalize both attacks to infer non-binary attributes.

5.4 Defenses

In this section, we briefly outline some of the approaches that may be used to mitigate the effects of our noise-exploitations attacks – and other attacks – against Diffix and other privacy-preserving query-based systems. Overall, it is our belief that practical secure design principles apply here just as they do in many other contents. Specifically, privacy-preserving query-based system such as Diffix (regardless of whether they have provable guarantees or not) would benefit from a defense-in-depth approach, by monitoring the query stream for queries that are likely to lead to exploitation.

Intrusion detection. The set of queries generated by our attacks follow a specific template. Learning this pattern may help prevent noise-exploitation attacks, as well as potentially related attacks. A more sophisticated attacker might however vary the shape of the queries and interleave them with other more natural-looking queries, including over long periods of time.

Auditability. If the user of such a system is authenticated, then a suspicious-looking query stream can lead to temporary account suspension and further investigations of their activity, including after the fact, as new attacks are being uncovered.

Increased friction. Another strategy involves imposing time delays or financial charges on queries, for instance by charging by the number of queries, instead of using a subscription-based model. This strategy can be refined to, for instance, charge more or create longer delays for suspicious queries. This would make it more difficult to automate the inference process at scale.

Limited expressiveness. Instead of a rich syntax, the mechanism could allow only for a small set of conditions that are easier to validate. This could also include a limit to the number of conditions per query, or to the total number of conditions that may be used by the authenticated system user during a specific interval of time. This involves a compromise between rough data summarization and fine-grained queries, and limits the utility of the system in practice.

5.5 Disclosure

After we discovered and prototyped our differential attack, we reached out to the authors of Diffix and shared with them our manuscript, which subsequently appeared on ArXiv.org. A week later, the authors of Diffix published a blog post on their website [30] discussing our results. While they acknowledge our attack, they claim that it is not practical as the necessary assumptions are rarely met in the datasets they analyzed.

We disagree with this claim. First, the existence of the attack, *independently* of the dataset, contradicts both the spirit and the letter of GDPR's Art. 29 WP. Second, we showed that, albeit correct, the authors' analysis was insufficient. In this paper, we give an example of a dataset on which the differential attack is very effective, even without an oracle. Moreover, we demonstrate that there exist real-world datasets on which the necessary assumptions are met for a significant fraction of users. Third, the cloning attack, which we developed after-

wards, is able to validate its assumptions automatically and performs very well on a large range of real-world datasets.

The code of our differential and cloning attacks, as well as the experiments performed in this paper, are available at https://cpg.doc.ic.ac.uk/signal-in-the-noise.

6 Related work

Attacks on query-based systems. Diffix is an example of query-based system: the individual-level (often pseudonymous) data is stored on the data curator's server. Users access the data exclusively by sending queries that only return information aggregated from several records. While this setup prevents traditional re-identification attacks [4–9], a large range of attacks on query-based systems have been developed since the late 70's [25, 31]. Most of these attacks show how to circumvent privacy safeguards (for instance, query set size restriction and noise addition) in specific setups. In 2003, Dinur et al. [32] proposed the first example of an attack that works on a large class of query-based systems. In what they called a reconstruction attack, they showed that if the noise added to every query is at most $o(\sqrt{n})$, where n is the size of the dataset, then an attacker can reconstruct almost the entire dataset using only polynomially many queries. Sankararaman et al. [33] realized the first formal study of tracing attacks, introducing a theoretical attack model based on hypothesis testing. While reconstruction attacks aim at inferring one or more attributes of some record in the dataset (violating the inference requirement of the Art. 29 WP), the goal of tracing attacks is only to determine whether the data about a certain individual (more precisely, their record) is present in the dataset. Numerous reconstruction and tracing attacks have been proposed in the literature. These attacks address different limitations of previous ones, particularly the computational time required to perform them. A recent survey from Dwork et al. [34] gives a detailed overview of attacks on query-based systems.

Attacks on differential privacy. Differential privacy is a privacy guarantee that can be enforced by query-based systems. Differential privacy has been mathematically proven to be robust against a very large class of attacks [12] when used with an appropriate privacy budget ε. However, research has shown that attacks on *implementations* of differentially private systems exist. We give an overview in the full version.

Differential privacy for general-purpose analytics. Diffix was specifically created as an alternative to differential privacy to provide a better privacy/utility tradeoff for general-purpose analytics [35, 36]. Specifically, Diffix allows for infinitely many queries with little noise added to outputs.

General-purpose analytics usually refers to systems that allow analysts to send many queries of different type, and ideally permit to join different datasets. Some solutions based on differential privacy have been proposed, the main ones being PINQ [37], wPINQ [38], Airavat [39], and GUPT [40]. All of these systems however present limitations, e.g. simplicity of use and support for various operators that join different datasets [13]. In 2017, Johnson et al. [13] proposed a new framework for general-purpose analytics, called FLEX, developed in collaboration with Uber. FLEX enforces differential privacy for SQL queries without requiring any knowledge about differential privacy from the analyst. However, the actual utility achieved – level of noise added – by the current implementation of FLEX has been questioned [41].

Attacks on data-dependent noise. Values of Diffix's dynamic noise for a query depend on the query set (i.e. the set of users selected by the query), and hence on the data. This is what allows for our noise-exploitation attack to work. Data-dependent noise, also called instance-based noise, has been shown to provide significantly better accuracy than data-independent noise [42]. However, naive implementations of data-dependent noise can leak information about the data, a result Nissim et al. theorized as a potential way to attack the system [42]. To the best of our knowledge, our noise-exploitation attack is the first instance of an attack exploiting specifically data-dependent noise on deployed systems.

6.1 Other attacks on Diffix

We published the first version of our paper on ArXiv.org in April 2018, describing the differential attack. We updated it with a cloning attack in July 2018. Two months later, in October 2018, two other attacks on Diffix were disclosed. A membership attack by Pyrgelis et al. [43], based on a previous paper [44], and a reconstruction attack by Cohen and Nissim [45], based on previous work by Dinur et al. [32] and Dwork et al. [46]. These attacks are very different from ours and require a large number of queries in a typical setting, while our cloning attack can work with only 32 queries (see Appendix B). These are, to the best of our knowledge, the only three attacks specifically targeting Diffix.

Membership attack on location data. The attack by Pyrgelis et al. [43] is as follows: the attacker trains a machine learning algorithm on a *linkability* dataset (the attacker's background knowledge) to infer the presence of a user in a *protected* dataset (accessible only through queries on Diffix). Both datasets contain the full trajectories of users and half of the users are present in both datasets. The classifier is trained on the linkability dataset and queries on Diffix that count the number of people transiting in a certain area at a given time. The experimental results focus on the top 100 users with the highest number of reported locations in the linkability dataset. Out of 62 users present in both datasets, the classifier correctly infers the presence for 50 of them.

This attack presents three limitations. First, it is a membership attack and only allows an attacker to infer whether a person is in the protected dataset or not. Second, it assumes a strong adversary who has access to the full trajectory of a user exactly as it exists in the protected dataset, for a large number

of users. Third, the attack requires about 32,000 queries to assess the presence of a user. Membership attacks are however very useful when combined with inference attacks like ours, allowing an adversary to effectively verify our assumption that the victim is in the dataset.

Linear program reconstruction attack. The attack by Cohen et al. [45] focuses on reconstructing the dataset. In its simplest form, the attack assumes that the dataset contains n records, and each user $i \in [n]$ has a binary attribute s_i. The attack then selects random subsets of users and, for each subset $I \subseteq [n]$, queries Diffix for the result of $\sum_{i \in I} s_i$. This allows the attacker to produce a noisy linear system that can be solved using linear programming techniques to reconstruct the entire set of secret attributes $\{s_1, \ldots, s_n\}$ with perfect accuracy in polynomial time.

While this attack can successfully reconstruct the entire dataset, it presents two limitations compared to our attack.

First, it requires that the system allows queries of the type $\sum_{i \in I} s_i$, i.e. queries that select any analyst-defined set of users $I \subseteq [n]$, the "row-naming problem". The authors here exploit SQL functions supported by Diffix to define hash functions which they then use to select "random enough" sets of users. Following the disclosure, Aircloak restricted the available SQL functions to prevent the attack [47].

Second, to target a specific user, the attack would require a number of queries proportional to the number of records. Since the attacker does not know *which* name $i \in [n]$ corresponds to the victim's record, it is necessary to fully reconstruct at least a few columns entirely. The attacker would then perform a uniqueness attack on the reconstructed dataset to infer the secret attribute of the victim.

On the contrary, the number of queries used by our attacks is independent of the number of records in the dataset.

7 Conclusion

The Diffix mechanism has recently been proposed as an alternative to data anonymization methods and differential privacy, and is currently used in production. The mechanism is claimed to allow an analyst to submit an unbounded number of queries, while thwarting inference attacks, as defined by EU's Art. 29 WP. In this paper, we show that Diffix's anonymization mechanism is vulnerable to a new class of attacks, which we call noise-exploitation attacks. Our attacks leverage design flaws in Diffix's data-dependent noise to infer private attributes of an individual in the dataset, solely from prior knowledge about other attributes of this individual. In our opinion, Diffix alone and in its present state likely fails to satisfy the EU's Art. 29 WP requirements for data anonymization. Furthermore, our results show that naive data-dependent noise leads to highly vulnerable systems.

Our differential noise-exploitation attack, given little auxiliary information about the victim, combines specific queries and estimates how the noise is distributed to infer the value of the private attribute. In a synthetic best-case dataset, the attacker can predict with 92.6% accuracy private attributes, using only 5 attributes.

Our cloning noise-exploitation attack extends the first one by adding "dummy" conditions that do not change the selected query set. It relies on weaker assumptions, that are automatically validated with high accuracy by our algorithm. We evaluate its performances on four real-world datasets and find that it infers private attributes of between 87.0% and 97.0% of all records across datasets.

We finally recommend four defense-in-depth principles to defeat the de-anonymization attacks we describe.

References

[1] David Lazer, Alex Sandy Pentland, Lada Adamic, Sinan Aral, Albert Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, and Others. Life in the network: the coming age of computational social science. *Science*, 323(5915):721, 2009.

[2] Carole Cadwalladr and Emma Graham-Harrison. Revealed: 50 million facebook profiles harvested for cambridge analytica in major data breach. *The Guardian*, 17, 2018.

[3] Council of European Union. Regulation (EU) 2016/679. *OJ*, L 119:1–88, May 2016.

[4] L Sweeney. Weaving technology and policy together to maintain confidentiality. *J. Law Med. Ethics*, 25(2-3):98–110, 82, 1997.

[5] A Narayanan and V Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. ieeexplore.ieee.org, May 2008.

[6] Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Sci. Rep.*, 3:1376, 2013.

[7] Yves-Alexandre de Montjoye, Laura Radaelli, Vivek Kumar Singh, and Alex "sandy" Pentland. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221):536–539, January 2015.

[8] Chris Culnane, Benjamin I P Rubinstein, and Vanessa Teague. Health data in an open world. *ArXiv e-prints*, December 2017.

[9] P Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA Law Rev.*, 2010.

[10] President's Council of Advisors on Science and Technology. Big data and privacy: a technological perspective. Technical report, White House, January 2014.

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. *Calibrating Noise to Sensitivity in Private Data Analysis*, page 265–284. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Mar 2006.

[12] Shiva P Kasiviswanathan and Adam Smith. On the 'semantics' of differential privacy: A bayesian formulation. *1*, 6(1), June 2014.

[13] Noah Johnson, Joseph P. Near, and Dawn Song. Towards practical differential privacy for sql queries. *ArXiv e-prints*, Jun 2017. arXiv: 1706.09479.

[14] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. Privacy loss in apple's implementation of differential privacy on macos 10.12. *ArXiv e-prints*, September 2017.

[15] Ilya Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 650–661, New York, NY, USA, 2012. ACM.

[16] Andreas Haeberlen, Benjamin C Pierce, and Arjun Narayan. Differential privacy under fire. In *USENIX Security Symposium*, 2011.

[17] Paul Francis, Sebastian Probst Eide, and Reinhard Munz. Diffix: High-Utility database anonymization. In *Privacy Technologies and Policy*, pages 141–158. Springer International Publishing, 2017.

[18] P. Francis, S. Probst-Eide, P. Obrok, C. Berneanu, S. Juric, and R. Munz. Extended Diffix. *ArXiv e-prints*, June 2018.

[19] Article 29 Data Protection Working Party. Opinion 05/2014 on anonymisation techniques. April 2014.

[20] Aircloak. Announcing our seed investment. https://web.archive.org/web/20180426131132/https://blog.aircloak.com/announcing-our-seed-investment-26f392f1068a, October 2017.

[21] Aircloak. Building trust. https://web.archive.org/web/20180426104935/https://blog.aircloak.com/building-trust-35c74424efc6, July 2017.

[22] William Stallings, Lawrie Brown, Michael D Bauer, and Arup Kumar Bhattacharjee. *Computer Security: Principles and Practice*. Pearson Education, 2012.

[23] Nabil R. Adam and John C. Worthmann. Security-control Methods for Statistical Databases: A Comparative Study. *ACM Comput. Surv.*, 21(4):515–556, December 1989.

[24] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: K-anonymity and its enforcement through generalization and suppression. Technical report, technical report, SRI International, 1998.

[25] Leland L. Beck. A security machanism for statistical database. *ACM Trans. Database Syst.*, 5(3):316–3338, Sep 1980.

[26] Center for Machine Learning and Intelligent Systems. Adult dataset. https://archive.ics.uci.edu/ml/datasets/adult.

[27] Center for Machine Learning and Intelligent Systems. Credit dataset. https://archive.ics.uci.edu/ml/datasets/Credit+Approval.

[28] Center for Machine Learning and Intelligent Systems. Census dataset. https://archive.ics.uci.edu/ml/datasets/Census-Income+(KDD).

[29] Frank McSherry. Statistical inference considered harmful. https://github.com/frankmcsherry/blog/blob/master/posts/2016-06-14.md, 2016.

[30] Aircloak. Report on the diffix vulnerability announced by imperial college london and cu louvain. https://aircloak.com/report-on-the-diffix-vulnerability-announced-by-imperial-college-london-and-cu-louvain, April 2018.

[31] Dorothy E Denning. Are statistical data bases secure. In *Proc. AFIPS*, volume 2978, pages 525–530, 1978.

[32] 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*, page 202–210. ACM, 2003.

[33] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. Genomic privacy and limits of individual detection in a pool. *Nature Genetics*, 41(9):965–967, Sep 2009.

[34] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4(1):61–84, Mar 2017.

[35] Paul Francis. Mydata 2017 workshop abstract: Technical issues and approaches in personal data management.

https://aircloak.com/mydata-2017-workshop-abstract-technical-issues-and-approaches-in-personal-data-management, 2017.

[36] Paul Francis. Diffix: Enabling (aggregate) data markets with anonymization. https://aircloak.com/wp-content/uploads/mydata-market-aug17.pdf, 2017.

[37] Frank D McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 19–30, New York, NY, USA, 2009. ACM.

[38] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *Proceedings VLDB Endowment*, 7(8):637–648, April 2014.

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

[40] Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. GUPT: Privacy preserving data analysis made easy. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 349–360, New York, NY, USA, 2012. ACM.

[41] Frank McSherry. Uber's differential privacy .. probably isn't. https://github.com/frankmcsherry/blog/blob/master/posts/2018-02-25.md, 2018.

[42] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 75–84, New York, NY, USA, 2007. ACM.

[43] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. On location, time, and membership: Studying how aggregate location data can harm users' privacy. https://www.benthamsgaze.org/2018/10/02/on-location-time-and-membership-studying-how-aggregate-location-data-can-harm-users-privacy/, October 2018.

[44] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock Knock, Who's There? Membership Inference on Aggregate Location Data. In *Proceedings 2018 Network and Distributed System Security Symposium*, San Diego, CA, 2018. Internet Society.

[45] Aloni Cohen and Kobbi Nissim. Linear Program Reconstruction in Practice. *TPDP 2018*, October 2018.

[46] Cynthia Dwork, Frank McSherry, and Kunal Talwar. The Price of Privacy and the Limits of LP Decoding. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 85–94, New York, NY, USA, 2007. ACM.

[47] Aircloak. Fix for the mit/georgetown univ attack on diffix. https://aircloak.com/fix-for-the-mit-georgetown-univ-attack-on-diffix, October 2018.

[48] G. A Young and Richard L Smith. *Essentials of statistical inference: G.A. Young, R.L. Smith.* Cambridge University Press, 2005.

[49] Munuswamy Sankaran. Approximations to the non-central chi-square distribution. *Biometrika*, 50(1–2):199–204, Jun 1963.

A Likelihood ratio test

Let X and Y be independent random variables. Suppose that we have two hypotheses about the distributions of X and:

$$\begin{aligned} H_0: &\quad X \sim \mathcal{N}(\mu_0, \sigma_0^2) \quad \text{and} \quad Y \sim \mathcal{N}(\mu_1, \sigma_1^2) \\ H_1: &\quad X \sim \mathcal{N}(\mu_1, \sigma_1^2) \quad \text{and} \quad Y \sim \mathcal{N}(\mu_0, \sigma_0^2) \end{aligned}$$

where $\mu_0, \mu_1, \sigma_0, \sigma_1$ are known and fixed values such that $\mu_0 < \mu_1$ and $\sigma_0^2 < \sigma_1^2$. $\mathcal{N}(\mu, \sigma^2)$ denotes the the normal distribution with mean μ and variance σ^2.

Suppose we have a vector of n realizations $\mathbf{x} = (x_1, \ldots, x_n)$ of X and a vector of n realizations $\mathbf{y} = (y_1, \ldots, y_n)$ of Y. We assume that all the $2n$ realizations are mutually independent. The standard frequentist way to accept the preferred hypothesis H_0 or refute it (in favor of H_1) would use a likelihood ratio test with a pre-defined confidence level from which to derive critical regions [48]. In our case we do not have a preferred hypothesis, and hence we define a slightly different test.

Let f and g denote the probability density functions of $\mathcal{N}(\mu_0, \sigma_0^2)$ and $\mathcal{N}(\mu_1, \sigma_1^2)$ respectively. We define the likelihood ratio function Λ as follows:

$$\Lambda(\mathbf{x}, \mathbf{y}) = \prod_{j=1}^{n} \frac{f(x_j)}{g(x_j)} \prod_{j=1}^{n} \frac{g(y_j)}{f(y_j)}.$$

We accept H_0 if $\Lambda(\mathbf{x}, \mathbf{y}) \geq 1$, and we accept H_1 if $\Lambda(\mathbf{x}, \mathbf{y}) < 1$.

Theoretical accuracy of the test. The test will sometimes yield the wrong result. It is possible to determine what is the probability that this happens. Such probability depends on mean and variance of the two specific normal distributions.

Fact. *Let* $p_{err_0} = \Pr[\Lambda(\mathbf{x}, \mathbf{y}) < 1 \mid H_0]$ *and* $p_{err_1} = \Pr[\Lambda(\mathbf{x}, \mathbf{y}) \geq 1 \mid H_1]$. *Then*

$$p_{err_0} = p_{err_1} = \Pr[\alpha_0 Z_0 - \alpha_1 Z_1 < 0]$$

where, for $i = 0, 1$, Z_i is a noncentral chi-squared distribution with n degrees of freedom and noncentrality parameter

$$\lambda_i = n\left(\frac{\mu_i}{\sigma_i} + \frac{\mu_0\sigma_1^2 - \mu_1\sigma_0^2}{\sigma_i(\sigma_0^2 - \sigma_1^2)}\right)^2$$

and

$$\alpha_i = \frac{\sigma_0^2 - \sigma_1^2}{2\sigma_{1-i}^2}.$$

To prove the fact, one considers the log-likelihood ratio function $\log\Lambda(\mathbf{x},\mathbf{y})$ and applies elementary algebra to the obtained expression to derive a linear combination of noncentral chi-squared distributions. We omit the details.

Since $p_{\text{err}_0} = p_{\text{err}_1}$, we refer to this quantity simply as p_{err}. The accuracy of the test is $\text{acc} = 1 - p_{\text{err}}$.

We now show how to apply the fact to our differential noise-exploitation attack. For simplicity, we suppose that Diffix's outputs are not rounded to the nearest nonnegative integer and bucket suppression is never triggered for the queries in the attack, so that every pair of queries $(\widetilde{Q}_j, \widetilde{Q}'_j)$ and $(\widetilde{R}_j, \widetilde{R}'_j)$ yields a valid sample. Thus, for k known attributes, we have two vectors of samples $\mathbf{q} = (q_1, \ldots, q_k)$ and $\mathbf{r} = (r_1, \ldots, r_k)$ and for every $j \leq k$:

$$q_j \sim \begin{cases} \mathcal{N}(0,2) & \text{if } x^{(s)} = 1 \\ \mathcal{N}(1,2k+2) & \text{if } x^{(s)} = 0 \end{cases}$$

$$r_j \sim \begin{cases} \mathcal{N}(1,2k+2) & \text{if } x^{(s)} = 1 \\ \mathcal{N}(0,2) & \text{if } x^{(s)} = 0 \end{cases}$$

We assume that the $2k$ samples in $\mathbf{q}$ and $\mathbf{r}$ are mutually independent. As discussed in section 3, this is not always guaranteed to be true, but it has close to no effect on the actual accuracy of the test. Let

$$\begin{aligned} H_0 &: \quad x^{(s)} = 1 \\ H_1 &: \quad x^{(s)} = 0. \end{aligned}$$

Let f and g denote the probability density functions of $\mathcal{N}(0,2)$ and $\mathcal{N}(1,2k+2)$ respectively. Observe that H_0 holds if and only if every $q_j \sim f$ and every $r_j \sim g$. Similarly, H_1 holds if and only if every $q_j \sim g$ and every $r_j \sim f$. Then we can apply the test defined above. Define

$$\Lambda(\mathbf{q},\mathbf{r}) = \prod_{j=1}^{k} \frac{f(q_j)}{g(q_j)} \prod_{j=1}^{k} \frac{g(r_j)}{f(r_j)}.$$

Our test concludes that $x^{(s)} = 1$ if $\Lambda(\mathbf{q},\mathbf{r}) \geq 1$, and $x^{(s)} = 0$ if $\Lambda(\mathbf{q},\mathbf{r}) < 1$.

To measure the theoretical accuracy of the attack for k known attributes, we can apply the fact to $\Lambda(\mathbf{q},\mathbf{r})$ with $\mu_0 = 0, \sigma_0^2 = 2, \mu_1 = 1, \sigma_1^2 = 2k+2$ and $n = k$, and finally find $\text{acc}(k) = 1 - p_{\text{err}}(k)$.

Fig. 2 shows the values of $\text{acc}(k)$ for increasing values of k. Computing the value of p_{err} requires an approximation of the cumulative distribution function of a linear combination of noncentral chi-squared distributions, for which an exact closed-form expression is not known [49]. We compute these values using the R package `sadists`[3] version 0.2.3.

Numerical simulation with rounding. If we suppose that Diffix's outputs are rounded to the nearest nonnegative integer, no simple expression can be determined for the error rate. To estimate the accuracy in this case, we numerically simulate the values of $\widetilde{Q}_j(D)$ and $\widetilde{Q}'_j(D)$ that would result from querying Diffix (without bucket suppression), for different values of the secret attribute $x^{(s)}$. We then obtain each sample as the difference of the rounded results:

$$q_j = \text{round}(\widetilde{Q}_j(D)) - \text{round}(\widetilde{Q}'_j(D)).$$

Finally, we perform the likelihood ratio test as for the continuous case (considering also null outputs) and check whether the result is correct. We use balanced truth values for $x^{(s)}$, and perform 1000 experiments (on different queries) for each value of $x^{(s)}$. The results are shown in Fig. 2.

B Reducing the number of queries

One of the main features of Diffix is that it allows analysts to send an unlimited amount of queries. Many privacy attacks work by issuing a relatively large number of queries (see also 6.1). Limiting the number of queries allowed by Diffix would thwart or significantly affect these attacks. While our actual attack procedures require a small number of queries ($2|\Delta| + 1$ for the cloning attack), the subset exploration step can sometimes explore many sets of attributes before finding an exploitable one. To minimize the number of queries, we replace the iterative exploration with a greedy heuristic that selects only one subset which is likely to work. We focus only on the cloning attack, as it does not require an oracle and achieves much better accuracy.

The cloning attack requires a set of attributes (A', u), where the restricted record $x^{(A',u)}$ uniquely identifies the victim, but the vector $x^{(A')}$ is shared across a larger population (to avoid bucket suppression). The FullCloningAttack starts with a larger set of attributes A^* and iteratively explores subsets of A^* to find a candidate (A', u). We replace this iterative process with a single deterministic step.

Intuitively, we want u to be as discriminative as possible, while for the attributes in A' to select as many users as possible. This is what the procedure GreedySelectSubset does. First, it computes the (approximate) fraction of users that share the same value $x^{(a)}$, for each attribute $a \in A^*$. Then it selects as u the attribute associated with the lowest fraction. Now suppose that N is the estimated total number of users in the dataset. The set A' is selected as the smallest set of attributes

[3] https://github.com/shabbychef/sadists

associated with the highest fraction, additionally requiring that the product of all the fractions for (A', u) is smaller than $1/N$. This ensures that, with high probability, the victim is uniquely identified by $x^{(A',u)}$.

Procedure GreedySelectSubset($A^*, x^{(A^*)}, s, v$)

Input: known attributes (names A^* and values $x^{(A^*)}$), secret s and target value v
Output: a set of attributes $(A', u) \subseteq A^*$

$N \leftarrow \mathsf{count}()$ `// approx. tot. number of users`
foreach $a \in A^*$ **do**
  $C_a \leftarrow \mathsf{count}(a = x^{(a)} \wedge s \neq v)$
  $\rho_a \leftarrow \frac{C_a}{N}$
end foreach
$\{\rho_1, \ldots, \rho_{|A^*|}\} \leftarrow \text{SortDescendingOrder}(\{\rho_a\}_{a \in A^*})$
$u \leftarrow a_{|A^*|}$
$i \leftarrow 1, \quad A' \leftarrow \emptyset$
while $\rho_u \prod_{a_i \in A'} \rho_{a_i} > \frac{1}{N}$ **do**
  $A' \leftarrow A' \cup \{a_i\}$
  $i \leftarrow i + 1$
end while
return (A', u)

We can modify the FullDifferentialAttack replacing the subset exploration with this heuristic. The modified full attack is described in the GreedyFullCloningAttack procedure.

Procedure GreedyFullCloningAttack($A^*, x^{(A^*)}, \Delta, s, v$)

Input: known attributes (names A^* and values $x^{(A^*)}$), dummy conditions Δ, secret s and target value v
Output: True if $x^{(s)} = v$, False if $x^{(s)} \neq v$

$(A', u) \leftarrow$ GreedySelectSubset($A^*, x^{(A^*)}, s, v$)
if NoBucketSuppression($A', u, x^{(A)}, \Delta, s, v$) and ValueUnique($A', u, x^{(A^*)}$) **then**
  return CloningAttack($A', u, x^{(A)}, \Delta, s, v$)
end if
return NonAttackable

Observe that the GreedySelectSubset procedure issues exactly $|A^*| + 1$ queries. The differential attack with the assumption validation step issues at most $2|\Delta| + 1$ queries. So, the GreedyFullCloningAttack algorithm requires at most $|A^*| + 2|\Delta| + 2$ queries.

Subset selection	**Median n. queries**	**Max n. queries**	**Predicted attackable**	**accuracy$_{pa}$**
Iterative	304	5310	96.8%	93.3%
Heuristic	32	32	55.4%	91.7%

Table 2: Empirical results of the cloning attack with iterative subset exploration and with the heuristic subset selection.

We compared the performances of the FullCloningAttack and GreedyFullCloningAttack on the ADULT dataset, with the salary class as secret attribute and the other 10 attributes as A^*. As in section 4, we used $|\Delta| = 10$ dummy conditions and ran the attack on 1000 random users. The results are summarized in Table 2.

The maximum (and median) number of queries used by GreedyFullCloningAttack for a single user is $10 + 2 \times 10 + 2 = 32$. The median number of queries used by GreedyFullCloningAttack is about 10 times higher, and the maximum is 100 times higher.

GreedyFullCloningAttack effectively attacks more than half of the users, as opposed to 96.8% of the users for the FullCloningAttack. This is due to the fact that the first attack tries a single subset of attributes per user. However, this figure is still remarkably high, given the huge reduction of required queries. Finally, the accuracy of the inference for the attacked users is almost the same.

We believe that these results give additional evidence of the power, extendability and practicability of our noise-exploitation attacks. Introducing additional optimizations, the accuracy could be improved and the number of queries could be further reduced (see full version).