Understanding ε for Differential Privacy in Differencing Attack Scenarios

Narges Ashena[1(✉)], Daniele Dell'Aglio[1,2], and Abraham Bernstein[1]

[1] Department of Informatics, University of Zurich, Zurich, Switzerland
{ashena,bernstein}@ifi.uzh.ch

[2] Department of Computer Science, Aalborg University, Aalborg, Denmark
dade@cs.aau.dk

Abstract. One of the recent notions of privacy protection is Differential Privacy (DP) with potential application in several personal data protection settings. DP acts as an intermediate layer between a private dataset and data analysts introducing privacy by injecting noise into the results of queries. Key to DP is the role of ε – a parameter that controls the magnitude of injected noise and, therefore, the trade-off between utility and privacy. Choosing proper ε value is a key challenge and a non-trivial task, as there is no straightforward way to assess the level of privacy loss associated with a given ε value. In this study, we measure the privacy loss imposed by a given ε through an adversarial model that exploits auxiliary information. We define the adversarial model and the privacy loss based on a differencing attack and the success probability of such an attack, respectively. Then, we restrict the probability of a successful differencing attack by tuning the ε. The result is an approach for setting ε based on the probability of a successful differencing attack and, hence, privacy leak. Our evaluation finds that setting ε based on some of the approaches presented in related work does not seem to offer adequate protection against the adversarial model introduced in this paper. Furthermore, our analysis shows that the ε selected by our proposed approach provides privacy protection for the adversary model in this paper and the adversary models in the related work.

Keywords: Differential privacy · Parameter tuning · Differencing attack

1 Introduction

Differential privacy (DP) has changed the very notion of how privacy preservation is being achieved. As such, it has been embraced by industry, governments, and the scientific community. At the core of DP mechanisms, lies the parameter ε, which regulates the amount of noise added to results. ε takes positive values, and as it increases the magnitude of the noise decreases, resulting in less private and more accurate outcomes. ε is, hence, the knob that controls the trade-off between utility and privacy. In a typical data publishing scenario, data owners desire high privacy and data analysts look for accurate results. Most of the times it is a owner/curator's task to set ε. It is usually straightforward to calculate the error bound that a given value of ε introduces on the data analyst side. However, understanding the role of ε in relation to the formal DP is challenging. After more than a decade since DP's introduction, the literature of DP is sparse in the area of understanding and setting ε. US Census Bureau emphasizes on this very issue for publishing 2020 census data under DP protection [6]. The main objective of this paper is to better support the data owner by defining a more intuitive way of sensing privacy leakages due to inappropriate ε selection. Only a few studies have moved initial steps towards this direction [7,11,13,15].

[13,15] propose interpretations of ε by studying the probability of privacy leak events within a differencing attack scenario. In such a scenario, there is an analyst with permission to investigate a private dataset through various aggregate query submissions. This analyst is an *adversary*, who performs DA (Differencing Attack) to discover the binary secret bit of participants of the dataset. A DA consists of submitting two aggregate queries aiming at the secret bit. The first query addresses a subset of the dataset, while the second addresses the same subset in addition to the target person. The difference between the two query answers can reveal the target person's secret bit. [13,15] assume that the dataset is public (also known by the adversary) except for the secret bits. [13] focuses on the cases where all the participants but one have the secret bit set to 1 (i.e., the privacy leak happens when the identity of the person with secret bit 0 is learnt by the adversary), while [15] targets scenarios where every participant has the secret bit set to either 0 or 1. [15] also assumes that the secret bit set to 1 is more critical than 0, e.g., as being positive to HIV$^+$.

In this paper, we study the role of *differencing attacks* in a scenario similar to the ones in [13,15], where the privacy threat is the leakage of participants' binary secret bit. However, in contrast to [13,15], we have two different assumptions: (a) an identical level of importance for both values of secret bit and (b) less public knowledge about the dataset. As a solution, we propose to tune ε according to the probability P_{SDA} of successfully performing (where SDA stands for Successful Differencing Attack) a differencing attack on the dataset to be protected.

Hereby, our contributions are:

- the introduction of P_{SDA}, a metric to measure a privacy leakage of a differencing attack under DP protection.
- a method to compute P_{SDA} for different values of ε and three query types: *count*, *sum*, and *average* queries.
- an experimental comparison of P_{SDA} with [13,15]. Our attack strategy performs as successful as [13] for *average* queries, and more successful in *sum* queries. Our approach achieves the same *recall* as [15] in detecting the secret when it is 1, and higher *accuracy* than [15] in detecting both secrets values.

The paper is organized as follows. Section 2 presents a case study that motivates this research. In Sect. 3, we introduce related work. Section 4 provides necessary background knowledge about DP. In Sect. 5, we elaborate on P_{SDA} and the method to compute it for three different aggregate queries. Section 6 provides practical application of P_{SDA} on real data. Section 7 compares the values for ε calculated by our method with state of the art methods. In the end, we present some final remarks and conclusion in Sects. 8 and 9 respectively.

2 Case Study

xyz.abc is an Internet Protocol Television (IPTV) company. xyz.abc users can watch on-air TV programs, choosing between a selection of channels from different European countries. xyz.abc stores users' demographics and viewership data, i.e., whether a user watches a specific channel or not. Table 1 shows a snippet of the xyz.abc dataset. Among viewers from Austria (AT), Alice, identified by the hash ID 3f1fb1c6, is the only English-speaking woman. This data is analyzed by xyz.abc data scientists to showcase to their stakeholders (e.g., advertisement companies) their analytic capabilities. Examples of analyses on viewers' data are aggregate queries such as *count*, *sum*, and *average*, which can be used to compute the market shares of the channels or program ratings.

Even if interested in aggregation queries, xyz.abc is worried about privacy leaks. Bob, an xyz.abc employee and acquaintance of Alice, wants to learn more about her by checking her records in the xyz.abc dataset. From discovering what Alice watches, Bob may obtain information about her political orientation by checking if she watches *Channel_1*, a conservative news channel. Setting up mechanisms to forbid queries about one specific user (as Alice) and allowing only for aggregate queries does not solve the problem. In this dataset, Alice is in a minority and she is vulnerable to differencing attack. Bob can perform such an attack by submitting two aggregate queries:[1]

Q_1^c. SELECT COUNT(HashID) FROM dataset
WHERE Gender = 'F' AND *Channel_1* = 1
Q_2^c. SELECT COUNT(HashID) FROM dataset WHERE Gender = 'F'
AND *Channel_1* = 1 AND (Language <> 'EN' OR Country <> 'AT').

The difference between the two queries exposes that Alice watched *Channel_1*, since the result of $Q_1^c - Q_2^c$ is 1. Even banning count queries is not enough, as Bob may decide to attack using other aggregate operators. For example, he could exploit his knowledge about the Alice's age:

Q_1^s. SELECT SUM(Age) FROM dataset WHERE Gender = 'F' AND *Channel_1* = 1
Q_2^s. SELECT SUM(Age) FROM dataset WHERE Gender = 'F' AND *Channel_1* = 1
AND (Language <> 'EN' OR Country <> 'AT').

[1] One may consider querying directly for Alice's record as
SELECT COUNT(HashID) FROM dataset
WHERE Gender = 'female' AND CHANNEL_1 = 1 AND
Language = 'English' AND Location = 'Austria'.
A Protected dataset may, however, not allow querying over a subset smaller than some threshold. The result is that most differecing attack scenarios in the literature only consider two queries.

Table 1. IPTV dataset of viewer ship. The provided row belongs to Alice. Channel_i is a binary attribute that indicates whether a user has watched the corresponding channel in a given time window or not.

HashID	Age	Gender	Language	Country	Device	Channel_1	Channel_2	...
...	...	F	DE	AT	...	...	...	...
...	...	M	DE	AT	...	...	...	...
...	...	F	DE	AT	...	...	...	...
3f1fb1c6	43	F	EN	AT	Screen	1	0	...
...	...	M	DE	AT	...	...	...	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

$Q_1^s - Q_2^s$ equals to Alice's age if Alice has watched *Channel*_1. For discovering Alice's secret, it suffices that the adversary finds out whether $Q_1 - Q_2 \neq 0$ or not regardless of the query type. One solution to neutralize the differencing attack is to release approximate results of the queries by noise addition, e.g., by using DP. Exploiting the approximate results, the adversary can not make a certain decision about the target person's secret. However, the adversary can have a guess and calculate the correctness probability of such a guess. Hence, the added noise has to be sufficiently large—ε has to be adjusted accordingly—to prevent an adversary from guessing the secret better than guessing randomly.

3 Related Work

Since DP was introduced, most of the research has focused on algorithm design and implementation of DP mechanisms [1,5,9,16,18–20]. Little attention has been dedicated so far to methods and solutions to set the parameter ε. Whilst there have been attempts in explaining DP to non-technical audience such as [22], to date, there are only four thorough studies on this topic [7,11,13,15].

In [22], the increase of probability of a harmful event in general is explained in terms of ε. In this study as well as [13], and [15], however, the probability of a leakage is investigated for a given ε and a specific attack scenario providing probabilities more attuned for these scenarios.

Lee, et al. [13] calculate an upper bound for ε to protect people's secret bit in a dataset. All members have the secret bit value set to 1 except for one individual, whose secret bit is 0. Their threat model assumes that the adversary knows every attribute about every individual, except for the binary secret attribute. The privacy leak happens when the adversary can infer with probability higher than a threshold value who is the person with secret 0. A proper value for ε does not allow the adversary to grow his/her success probability beyond such a threshold. Our method shares the idea of defining privacy loss as the probability of a successful attack while relaxing the assumptions on adversarial knowledge and presence of only one secret attribute value set to 0.

In [15], leakage happens when the adversary infers whether the data of a person contributes into the result of a query or not. This translates to secret bit set to 1 when that person's data is a factor in the query's output and 0 otherwise. To guess the secret bit, the adversary model utilizes the Neyman-Pearson hypothesis testing. It assumes that the risk of the secret equals to one is higher than the case where it is zero. This assumption is aligned with Neyman-Pearson criterion which aims to maximize the true detection rate by putting a constrain on false alarm rate and is valid in many disease-related scenarios. However, it does not apply to the cases where both secret values have the same significance, such as protecting participants' political beliefs (being left or right). In contrast, our proposed approach treats both values of secret equally.

In [7], Hsu et al. propose a solution based on economics for setting ε. They consider the case, where sensitive datasets are associated to a given amount of budget. Participants are compensated from that budget in case of privacy leakages. By assessing the risks of participating and calculating the expected cost for each participants, they calculate the upper bound for ε given the available budget, the desired accuracy, and the number of participants.

[11] takes a different approach for monetizing privacy. Here, data analysts provide privacy as a premium service to data owners. Data owners pay the analysts for their desired level of privacy, and each individual specifies a desired privacy level. Considering users' preferences and the analyst's requested accuracy level, the proposed mechanism calculates the differentially private query answers.

4 Preliminaries

In this section, we present the foundations on DP.

Differential Privacy. Let D be a dataset with size $n \geq 1$ records, where each record represents an individual and every column is an attribute. Neighboring datasets are defined as two datasets D and D' that have the same attributes and differ in one record, i.e., $|D - D'| = 1$. Let $\mathcal{D}$ be the set of datasets. Formally, a query $Q : \mathcal{D} \rightarrow \mathbb{R}$ processes a dataset and outputs a real answer. $\mathcal{M}$ is a randomized function that obfuscates the result of Q. Formally, a mechanism $\mathcal{M}$ is ε-differentially private if the inequality:

$$P(\mathcal{M}(D) \in S) \leq e^{\varepsilon} \cdot P(\mathcal{M}(D') \in S) \tag{1}$$

holds for every $S \subseteq range(Q)$ and for every pair of neighboring datasets. ε provides the mean to quantify the imposed privacy.

Composition Theorem. The composition theorem states that if multiple differentially private mechanisms access a dataset, the union of the outputs of these mechanisms is differentially private, and the privacy guarantee is the summation of the ε's of the applied mechanisms. Formally, if there are n mechanisms $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$ with ε equal to $\varepsilon_1, \ldots, \varepsilon_n$ respectively, the set of all these mechanisms $\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_n)$ is $\sum_{i=1}^{n} \varepsilon_i$-differentially private.

Laplace Mechanism and Global Sensitivity. One common way to build a differentially private mechanism is to add random noise drawn from a Laplace distribution with parameter $\frac{\Delta Q}{\varepsilon}$ to the query's result [4]. ΔQ is the *global sensitivity* of the query Q and is defined as:

$$\Delta Q = \max_{\forall D, D' \in \mathcal{D}: |D - D'| = 1} |Q(D) - Q(D')|. \tag{2}$$

In this paper, we consider three different queries: *count*, *sum*, and *average* queries.

In *count queries*, e.g., Q_1^c in Sect. 2, ΔQ is 1 because adding (or removing) a record in (from) a dataset can change the result at most by one.

In *sum queries*, ΔQ is the maximum value of the attribute, provided that the range of such an attribute is known. Datasets usually have several numerical attributes, each with its own range. One can either calculate ΔQ for each attribute separately, or normalize the attribute values between zero and one and consider $\Delta Q = 1$. In the following, we consider the latter case without loss of generality. Applying DP to *sum* queries on normalized data is similar to the count query case: it adds random noise drawn from $lap(\frac{1}{\varepsilon})$ to the real query result.

To compute differentially-private *average queries*, one naive solution is to compute the ratio between noisy *sum* and noisy *count* values. According to composition theorem, for this solution, the allocated ε for *average* query needs to be divided between *sum* and *count* causing noisier result than adding noise once according to full allocated ε. Alternatively, Algorithm 2.3 in [14] introduces the division of noisy sum by true count as a differentially private *average* with higher accuracy compared to the former solution for the same level of privacy. As shown in Algorithm 2.3 in [14], this division is not differentially private unless the result is restricted to the range of the attribute $[a_{min}, a_{max}]$.

Adding noise according to the global sensitivity of *average* query is another solution to compute differentially private average. To determine the *average* global sensitivity, we need to investigate the impact of adding (or removing) a value to (or from) a dataset. Let m be the average of the list $(a_1, \ldots, a_k, \ldots, a_n)$ where $a_i \in [a_{min}, a_{max}]$. Adding a new element a_{n+1} (in the range $[a_{min}, a_{max}]$) or removing an element a_k, changes m by:

$$\frac{a_{n+1} - m}{n+1} \tag{3}$$

or

$$-\frac{a_k - m}{n-1} \tag{4}$$

respectively. The global sensitivity is the maximum of the Eq. 3 and Eq. 4 values. For these two equations, the denominator is smallest when either $n = 1$ or $n - 1 = 1$. As mentioned earlier, we assume that datasets contain at least one record, $n \geq 1$. The maximum value of Eq. 3 happens where $n = 1$ and $a_{n+1} = a_{max}$ and $m = a_{min}$. For Eq. 4, the maximum value happens when $n - 1 = 1$ and $a_k = a_{max}$ and $m = \frac{a_{max} + a_{min}}{2}$. Therefore, the global sensitivity of *average* is $\frac{a_{max} - a_{min}}{2}$. The noise drawn from $lap(\frac{a_{max} - a_{min}}{2\varepsilon})$ is excessive for the accuracy of average and unnecessarily large since when the dataset size is sufficiently large, the contribution of a value in the average is considerably smaller than $lap(\frac{a_{max}-a_{min}}{2})$. For instance, the size of the dataset introduced in Sect. 2 can be 100 to 1000 or even larger. Thus, in this paper, to compute differentially private average, we consider Algorithm 2.3 in [14].

5 Our Approach to Interpreting ε

An ideal value for ε should minimize the privacy risks whilst reducing the noise introduced into the results of queries to ensure as adequate as possible answers. In the case of differencing attacks, this happens when an adversary cannot make a guess about the individuals' binary secret better than a random guess, i.e., success probability equal to 0.5. Our goal is, therefore, to find the largest value of ε that restricts the success of a differencing attack to 0.5. In the following, we first discuss assumptions on the adversary's prior knowledge for initiating a differencing attack. Next, we define P_{SDA} as the success probability of such an attack and calculate the approximation of P_{SDA}. Finally, we use P_{SDA} to set ε.

5.1 The Adversary's Prior Knowledge

DP literature often considers the worst case when it comes to the adversary's knowledge about a dataset, i.e., s/he is aware of all the records and attributes in the dataset, except for the secret bit of a target person. This is not a realistic assumption in real-world datasets with hundreds of thousand of records. Although the attack defined in [13] highly depends on this worst case scenario, for the differencing attack in this paper, the adversary needs to be aware of enough auxiliary information to successfully formulating Q_1 and Q_2 as exemplified in Sect. 2. This information may be easily gathered from knowing the true result of only one histogram query over the dataset. This point is worth emphasizing as being focused on worst case scenario may mislead data owners about the safety of their datasets. If a data owner presumes none of the subsets of the dataset with full amounts of detail are publicly accessible, s/he may believe that the attacks introduced in [13,15] are ineffective. However, there can be other attacks relying on less knowledge than all of the records or every attribute of the dataset.

To conduct a successful attack, the adversary needs to compute d, which is the difference of the true values of Q_1 and Q_2, i.e., $Q_1 - Q_2$, when the target person's secret is not zero. According to the person's secret, $Q_1 - Q_2$ is either 0 or d. d varies according to the query type. For *count* queries, d is constant and equals to 1. In *sum* and *average* queries, d varies and depends on the data. Let a be the value of the attribute A of the target user. d equals a in the case of *sum* queries. For *average*, d is either $\frac{a-Q_2(D)}{n+1}$ or $\frac{a-Q_1(D)}{n-1}$ (as in Eq. 3 and Eq. 4).

Once the adversary defines Q_1 and Q_2 and calculates d, s/he interacts with the DP-protected dataset to receive noisy results of Q_1 and Q_2. For this interaction, the adversary is granted a fixed privacy budget ε. In this paper, we assume the adversary uses up the whole allocated privacy budget to perform a differencing attack. According to the composition theorem, the ε should be split between Q_1 and Q_2. We assume the ε is divided equally between the two queries.

Finally, we assume that the parameter b of the Laplace distribution employed by the DP mechanism is also public information. As shown in Sect. 4, b equals to $\frac{\Delta Q}{\varepsilon}$. There are many possibilities that this information becomes available for the adversary. To give insights on how to interpret the noisy results, DP based query systems often release the accuracy of the results, which are calculated according to the distribution used to generate the injected noise [4,5]. The reported accuracy along with the allocated ε can be used by the adversary to calculate $b = \frac{\Delta Q}{\varepsilon}$. Moreover, since most DP libraries and tools are open source, the mechanisms' implementations can be assumed to be publicly available.

5.2 Guessing the Secret

In this section, we discuss how to perform the differencing attack in a DP framework. In contrast to the attack described in Sect. 2, here the adversary receives $\mathcal{M}_1(D)$ and $\mathcal{M}_2(D)$, which are noisy versions of Q_1 and Q_2. S/he knows that:

$$Q_1(D) - Q_2(D) \in \{0, d\}. \tag{5}$$

As explained in Sect. 4, $\mathcal{M}_1(D) = Q_1(D) + \nu_1$ and $\mathcal{M}_2(D) = Q_2(D) + \nu_2$, where ν_1 and ν_2 are random samples drawn from $lap(\frac{2\Delta Q}{\varepsilon})$. The number two in the numerator comes from splitting the ε between the two queries. Hence, to infer the secret, the adversary takes the difference of the observations:

$$\mathcal{M}_1(D) - \mathcal{M}_2(D) = Q_1(D) + \nu_1 - (Q_2(D) + \nu_2). \tag{6}$$

By taking Eq. 5 into account, Eq. 6 simplifies to either Eq. 7 or Eq. 8.

$$\mathcal{M}_1(D) - \mathcal{M}_2(D) = \nu_1 - \nu_2 + 0 = \Delta\nu + 0 \tag{7}$$

$$\mathcal{M}_1(D) - \mathcal{M}_2(D) = \nu_1 - \nu_2 + d = \Delta\nu + d \tag{8}$$

The adversary knows $\mathcal{M}_1(D)$ and $\mathcal{M}_2(D)$, but not the values of ν_1, ν_2, or even $\Delta\nu$. However, s/he knows the probability density of $\Delta\nu$ as it is a linear combination of two Laplace distributions [10] with parameters $\frac{2\Delta f}{\varepsilon}$ and $\mu = 0$. Therefore, the adversary investigates which of the occurrences of Eq. 7 and Eq. 8 are more likely using the distribution of $\Delta\nu$. Let the $pdf_{\Delta\nu}$ be the probability density function of $\Delta\nu$. The adversary's guess for the secret bit of the target person is:

$$guess = \begin{cases} 1, \text{if } pdf_{\Delta\nu}(\mathcal{M}_1(D) - \mathcal{M}_2(D) - 0) < \\ \qquad pdf_{\Delta\nu}(\mathcal{M}_1(D) - \mathcal{M}_2(D) - d) \\ 0, \text{otherwise.} \end{cases} \tag{9}$$

The attack strategy explained above is applicable for *count* and *sum* queries but it requires slight changes for *average* queries. As mentioned in Sect. 5.1, to calculate d for the *average* query, in addition to the target person's attribute value a, the adversary requires some information about the dataset, e.g., the true result of either Q_1 or Q_2. As Q_1 is a more general query than Q_2, we assume without loss of generality Q_1 is public and known to the adversary.

Algorithm 1. Performs the differencing attack and guesses the target person's secret bit for given ε, DP results, and true result of either Q_1 or Q_2 if available

```
1:  function GUESS_SECRET( $\mathcal{M}_1$, $\mathcal{M}_2$, $\varepsilon$, $\Delta Q$, $d$,$Q_{1or2} = 0$, $n = 0$)
2:      $guess \leftarrow 0$
3:      if $Q_{1or2}! = 0$ then
4:          compute the $pdf_{lap}$ of $lap(\frac{\Delta Q}{\varepsilon})$
5:          if $\mathcal{M}_1! = 0$ then
6:              $\Delta\mathcal{M} \leftarrow \mathcal{M}_1 - Q_{1or2}$
7:          else
8:              $\Delta\mathcal{M} \leftarrow Q_{1or2} - \mathcal{M}_2$
9:          if $n == 0$ and $pdf_{lap}(\Delta\mathcal{M} - d) \geq pdf_{lap}(\Delta\mathcal{M} - 0)$ then
10:             $guess \leftarrow 1$
11:         if $n! = 0$ and $pdf_{lap}((n-1) \times (\Delta\mathcal{M} - d)) \geq pdf_{lap}(n \times (\Delta\mathcal{M} - 0))$ then
12:             $guess \leftarrow 1$
13:     else
14:         compute the $pdf$ of $\Delta\nu$ which is $lap(\frac{2\Delta Q}{\varepsilon}) - lap(\frac{2\Delta Q}{\varepsilon})$
15:         $\Delta\mathcal{M} \leftarrow \mathcal{M}_1 - \mathcal{M}_2$
16:         if $pdf_{\Delta\nu}(\Delta\mathcal{M} - d) \geq pdf_{\Delta\nu}(\Delta\mathcal{M} - 0)$ then
17:             $guess \leftarrow 1$
18:     return $guess$
```

Therefore, s/he only queries for Q_2 without the need to divide the allocated ε. In this case, Eq. 7 and Eq. 8 become $Q_1(D) - \mathcal{M}_2(D) = 0 - \frac{\nu}{n}$ and $Q_1(D) - \mathcal{M}_2(D) = \frac{a - Q_1(D)}{n-1} - \frac{\nu}{n-1}$ respectively. Using pdf_ν, which is basically the *pdf* of a Laplace distribution, the adversary makes a guess for the secret bit of the target person by comparing

$$pdf_\nu(n \times (Q_1(D) - \mathcal{M}_2(D) - 0)) \tag{10}$$

and

$$pdf_\nu((n-1) \times (Q_1(D) - \mathcal{M}_2(D) - \frac{a - Q_1(D)}{n-1})). \tag{11}$$

If Q_1 equals to a, it is not possible to guess the secret bit with probability more than 0.5 (even in case of no noise addition) as adding or removing a value to the dataset equal to the average does not affect the average. The same happens when the noisy average falls out of $[a_{min}, a_{max}]$ and the mechanism used for average returns a_{min} or a_{max} (see Sect. 4). Without the trace of added noise, the adversary can only guess randomly.

Algorithm 1 shows *GUESS_SECRET*. When the adversary is aware of the true result of either Q_1 or Q_2, the input Q_{1or2} equals to that true result. Depending on the known query, the corresponding $\mathcal{M}$ is set to zero.

5.3 The Definition of P_{SDA}

In this section, we define P_{SDA}—a metric for the probability of a successful differencing attack. Let Z be a random variable:

$$Z = \alpha A + \beta B, \tag{12}$$

where α and β are real numbers and A and B are two random variables. When the attack bases on the difference between the two queries, A and B are sampled from $lap(\frac{2\Delta Q}{\varepsilon})$, and α and β are set to one and minus one, respectively. When Q_1's real value is known (e.g., for the average query), α becomes zero and B is a Laplace distribution with parameter $\frac{\Delta Q}{\varepsilon}$.

The probability density function of Z is $pdf_Z(Z; \varepsilon, \Delta Q)$. pdf_Z is a function of Z defined based on ε and ΔQ. For an elaborate calculation of pdf_Z, we refer to Theorem 2 in [17]. Let g be a step function defined as:

$$g(y) = \begin{cases} 1, \text{if } y \geq 0 \\ 0, \text{if } y < 0 \end{cases} \tag{13}$$

The binary random variable X describes whether the *guess* from Eq. 9 is correct or not. X takes 1 if $guess = secret$ and 0 otherwise. Using equations Eq. 7, Eq. 8, and Eq. 9, X is defined as the XNOR between *guess* and actual *secret*:

$$X = 1 - |g(pdf_Z(\Delta\mathcal{M}; \varepsilon, \Delta Q) - pdf_Z(\Delta\mathcal{M} - d; \varepsilon, \Delta Q)) - secret| \tag{14}$$

We define P_{SDA} as the probability parameter of the binary random variable X. As the probability parameter of any binary random variable with zero and one values is its expected value, it follows that $P_{SDA} = P(X = 1) = E[X]$.

When the adversary guesses the target person's secret, s/he knows that the guess is correct with probability P_{SDA}. Intuitively, this translates to a scenario where the adversary owns a biased coin with probability P_{SDA}. Given a guess, s/he tosses the coin and decides whether such a guess is correct based on the outcome. With a heavily unbiased coin, the adversary's becomes more confident about the guess. Therefore, the risk of target person's secret leakage increases.

5.4 The Estimation of P_{SDA}

The calculation of P_{SDA} is not trivial because X, defined in Eq. 14, is a random variable defined as a combination of other two random variables. Therefore, we provide a method to calculate the approximated P_{SDA} called $\widehat{P}_{SDA}$. Let us assume that *secret* is known. To estimate $\widehat{P}_{SDA}$ with a tight confidence interval, the adversary needs to sample X for a sufficiently large number of times. This means that the adversary has to resubmit the queries many times, which would require way more budget than the one that is usually allowed.

Therefore, we take a different approach and *simulate* the interaction with the DP mechanism: we run this simulation long enough to assess $\widehat{P}_{SDA}$. Since our attack is designed without dependencies on the underlying data, there are no direct interactions with the dataset D. Furthermore, in this work, it is assumed that the adversary has sufficient computational resources to run the simulation with enough iterations.

As shown in Algorithm 2, for each iteration, a random independent value is assigned as the secret bit *secret* and two random samples are drawn from $lap(\frac{2\Delta f}{\varepsilon})$ as ν_1 and ν_2 (or just a single sample from $lap(\frac{\Delta f}{\varepsilon})$ in case of non-zero Q_{1or2}). The adversary makes a guess *guess* based on this potential observation $\nu_1 - \nu_2 + secret \times d$. In real interaction with DP mechanism, the adversary's guess would be based

Algorithm 2. Estimates P_{SDA} with confidence interval no bigger than CI for α confidence level

```
1: function ESTIMATE_$P_{SDA}$($\varepsilon$, $\Delta Q$, $d$, $Q_{1or2} = 0$, $n = 0$)
2:     initialize $CI$ and $\alpha$
3:     initialize $Q_{Idx}$ based on $Q_{1or2}$        ▷ $Q_{Idx}$ is 1 if $Q_1$ is provided and 2 otherwise
4:     $itr \leftarrow 0$
5:     $successes \leftarrow 0$
6:     while $true$ do
7:         $itr \leftarrow itr + 1$
8:         $secret \leftarrow faircoinflip$        ▷ secret is 1 if coin flips Head and 0 otherwise
9:         if $Q_{Idx} == 0$ then
10:            $\nu_1$ and $\nu_2 \leftarrow$ random samples from $lap(\frac{2\Delta Q}{\varepsilon})$
11:            $\mathcal{M}_1 \leftarrow \nu_1 + secret \times d$
12:            $\mathcal{M}_2 \leftarrow \nu_2$
13:        if $Q_{Idx} == 1$ then
14:            $\nu_2 \leftarrow$ random sample from $lap(\frac{\Delta Q}{\varepsilon})$
15:            $\mathcal{M}_1 \leftarrow 0$
16:            if $n == 0$ then
17:                $\mathcal{M}_2 \leftarrow Q_{1or2} + \nu_2 - secret \times d$
18:            else
19:                $\mathcal{M}_2 \leftarrow Q_{1or2} + \frac{\nu_2}{n - secret} - secret \times d$
20:        if $Q_{Idx} == 2$ then
21:            $\nu_1 \leftarrow$ random sample from $lap(\frac{\Delta Q}{\varepsilon})$
22:            if $n == 0$ then
23:                $\mathcal{M}_1 \leftarrow Q_{1or2} + \nu_1 + secret \times d$
24:            else
25:                $\mathcal{M}_1 \leftarrow Q_{1or2} + \frac{\nu_1}{n + secret} + secret \times d$
26:            $\mathcal{M}_2 \leftarrow 0$
27:        $guess \leftarrow GUESS\_SECRET(\mathcal{M}_1, \mathcal{M}_1, \varepsilon, \Delta Q, d, Q_{1or2})$
28:        if $guess == secret$ then
29:            $successes \leftarrow successes + 1$
30:        $P_{SA}, CI_a, CI_b \leftarrow binomial\_fitting(itr, sucess, \alpha)$
31:        if $CI_b - CI_a \leq CI$ then
32:            return $P_{SDA}$
```

on $\mathcal{M}_1(D) - \mathcal{M}_2(D)$. In the simulation, it is not possible to interact with the dataset. Thus, as $\mathcal{M}_1(D) - \mathcal{M}_2(D)$ is equal to $\nu_1 - \nu_2 + secret \times d$, we use the latter. Finally, if $guess$ equals to $secret$, this iteration is counted as a success. We calculate $\widehat{P}_{SDA}$ by counting the successes and fit a binomial distribution to this number. The parameter of the fitted distribution is $\widehat{P}_{SDA}$.

For binomial fitting, we use the Clopper-Pearson method [2]. This method takes three inputs: the number of iterations, the number of successful guesses, and the confidence level for the fit as α. It outputs the binomial mean along with a lower bound CI_a and an upper bound CI_b. The estimated mean falls in the interval defined by CI_a and CI_b with probability of $1 - \alpha$. In Algorithm 2, the whole procedure iterates until the interval for the estimated binomial fit agrees with the defined CI which is the length of the interval $[CI_a, CI_b]$.

5.5 Setting ε

An ideal value for ε should not allow P_{SDA} to exceed 0.5. This can be formulated as Eq. 15. As we use an approximation for P_{SDA}, the equality in Eq. 15 turns to an inequality in Eq. 16 where δ is a sufficiently small number ensuring the closeness of the approximation of P_{SDA} to its true value.

Algorithm 3. Calculates the biggest possible ε for given P_{SDA} for the given dataset D, $query_t ype$, and upper bound for tolerable P_{SDA}

```
1: function SET_$\varepsilon$( $D$, $query\_type$, target_$P_{SDA}$ $= 0.5$)
2:     based on $query\_type$, calculate $\Delta Q$
3:     based on $query\_type$ and $D$, calculate $d$
4:     decide on $Q_{Idx}$, $Q_{1or2}$, and $n$
5:     $\varepsilon_b \leftarrow$ upper bound on $\varepsilon$
6:     while $\varepsilon_b > 0$ do
7:         if target_$P_{SDA}$ $-\delta \leq ESTIMATEP\_P_{SDA}$ $(\varepsilon_b, \Delta Q, d, Q_{Idx}, Q_{1or2}, n) \leq$ target_$P_{SDA}$ $+\delta$ then
8:             return $\varepsilon_b$
9:         else
10:            $\varepsilon_b \leftarrow decrease \varepsilon_b$
```

$$\begin{aligned} &maximize\ \varepsilon \\ &subject\ to:\ P_{SDA} = 0.5 \end{aligned} \tag{15}$$

$$\begin{aligned} &maximize\ \varepsilon \\ &subject\ to:\ |P_{SDA} - 0.5| \leq \delta \end{aligned} \tag{16}$$

Algorithm 3 provides a method for setting ε for different query types. The function set_ε takes the most tolerable value for $\widehat{P}_{SDA}$ (in this paper 0.5) as $target\_P_{SDA}$, along with the dataset and query type. Given that the provided attack strategy aims at a specific individual, we need to identify and protect individuals that are at highest risk of exposure to protect the dataset. Therefore, at Line 3 of Algorithm 3, the calculation of d can be based on such an individual. One conservative way to initialize d is to set it to the maximum possible value ΔQ. At Line 4, the data owner needs to decide on whether adversary has access to the true result of Q_1 or Q_2. Finally, the algorithm starts with a sufficiently large ε. It explores for an upper bound by decreasing ε at every step and investigating whether the current ε provides the given P_{SDA} or not. The algorithm stops when the condition in Eq. 16 is met when $|\widehat{P}_{SDA}$- target_P$_{SDA}| \leq \delta$.

6 Evaluations

In this Section, we illustrate the relation between P_{SDA} and ε for the case study described in Sect. 2. The dataset used for this section is from a real IPTV provider. A considerable number of individuals in this dataset are vulnerable to be isolated with two queries due to the wide range of features and viewership captured by the company. However, for privacy concerns, throughout this section, we focus on an imaginary case of Alice. The results elaborated in this section are generated based on the Algorithm 2 with ε in the range $[0.01, 20]$ with step 0.01. We set the inputs to the function $ESTIMATE\_P_{SDA}$ i.e., d, Q_{1or2} and n according to the query type. δ in Eq. 16 is set to 0.01. Finally, for the Clopper-Pearson method, we set $\alpha = 0.01$ and $CI = 0.02$ as in [2].

It is worth noting that the purpose of the IPTV case study is to contextualize the problem. Therefore, none of the evaluation in this section is anyway dependent on the IPTV dataset.

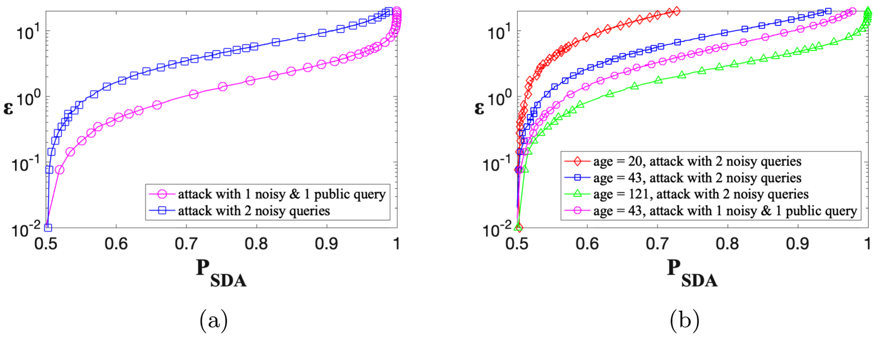

Fig. 1. P_{SDA} for (a) *count* and (b) *sum* queries for different values of ε (Color figure online)

Count Queries. To guess Alice's secret using a *count* query, Bob runs Q_1^c and Q_2^c as described in Sect. 2. Blue line in Fig. 1a represents how ε changes according to P_{SDA}. As ε grows, Bob can correctly guess the Alice's secret with higher probability. When the real value of Q_1^c is known (e.g., it is publicly available), the adversary has a higher chance in guessing the secret as shown by magenta line in Fig. 1a. To set ε for *count* query, once the data owner decides on publicity of Q_1^c or Q_2^c and tolerable P_{SDA}, ε can be selected.

Sum Queries. Figure 1b represents the case where the attacker is aware of the exact value of a queryable attribute of the target individual and exploits it to reveal a secret. The attacker decides to use a *sum* query for the attack. He submits the two queries Q_1^s and Q_2^s, as defined in Sect. 2. Differently from the count case, the attribute used in aggregations affects the leakage of the secret. The larger the attribute value, the more vulnerable the secret. Figure 1b, considers various hypothetical values of the age attribute. The red, blue, and green lines indicate the age of 20, 43 (Alice's real age), and 121[2] consecutively. It should be noted that the age of 121 is the maximum present in the dataset. Note that for normalized data (i.e., fitted between 0 and 1), the corresponding graph behaves exactly like the one shown for count query. Like in the count case, it can happen that the adversary knows the real value of one of the two queries. Magenta line in Fig. 1b depicts how the adversary success probability varies due to the auxiliary information.

An important question raises here: *what should the criteria for setting ε for sum query be?* Some may prefer to set ε based on the worst case, treating sum queries like count queries. However, this approach may add more noise than needed. For instance, in the dataset in Sect. 2, users with maximum age values are not at the risk of isolation with two queries. Hence, data owners may set the ε value based on those users' having the potential of being isolated.

[2] Many users tend to select the year 1900 as their birth year. Whilst this is unlikely to be the actual birth year of an user, we assume that this entry is still worthy of hiding, as it exposes an habit of the target person.

Average Queries. Let us assume that count and sum queries are not available. Hence, an adversary may exploit *average* queries, e.g., Bob wants to discover the secret of Alice through the following two queries Q_1^a and Q_2^a:

Q_1^a. SELECT AVG(Age) FROM dataset WHERE Gender = 'female' AND *Channel_1* = 1

Q_2^a. SELECT AVG(Age) FROM dataset WHERE Gender = 'female' AND *Channel_1* = 1 AND (Language <> 'English' OR Location <> 'Austria').

To perform the attack with *average* query, the adversary needs to know the value of the queryable attribute associated to the target individual (Alice's age), the size of the dataset (i.e., the real value of Q_1^c or Q_2^c as defined in Sect. 2), and the real value of Q_1^a or Q_2^a. Q_1^c and Q_1^a are both generic queries. It is possible for an adversary to collect the required information from the company reports or announcements. Therefore, we assume the adversary knows Q_1^a's true result.

Unlike the case of *sum* queries, where the magnitude of individual value (e.g. age) plays an important role in the secret leakage, here the distance between the individual value and the average value influences the leakage. The size of the dataset is another factor that may affect the secret disclosure. Figure 2a depicts P_{SDA} for three different ages and three different dataset sizes. We consider two hypothetical Alice's ages, 20 and 121, in addition to her true age, 43. As dataset sizes, we consider 13, 1320 and 16959 by limiting the *average* query to the number of users (including Alice) watched *Channel_1* in 1 h, 6 h, and 24 h consequentially. The average value of age is 50 in every case.

Figure 2a shows that the larger the distance of the true age from mean value, the higher the probability of a successful attack. As expected, due to the employed mechanism to release noisy *averages* in this paper, the size of dataset has a minor impact on the results. To simplify comparison of the P_{SDA} and ε for all three query types, Fig. 2b shows how Alice, 43 year old, is vulnerable to differencing attack in terms of P_{SDA} through her age. As shown in the figure, it is clear that privacy loss is different depending on the query type.

7 Comparison with Related Work

In this section, we qualitatively and quantitatively compare P_{SDA} with [13,15].

Comparing P_{SDA} with Lee et al., 2011. To compare P_{SDA} with Lee et al. [13], we shorty summarize the notions and assumptions taken in the latter. The dataset S includes two columns: the ID and a numerical attribute ($|S| = n$). S is fully known to the adversary. However, there is another dataset S', which is hidden from the adversary ($S' \subset S$ and $|S| = |S'| + 1$). The adversary's objective is to ascertain who is excluded in S'. Therefore, the absence from S' is the secret to be protected. The adversary assigns a uniform prior ($\frac{1}{n}$) to all the n possible subsets S'_i $(i = 1, ..., n)$ of S such that $|S| = |S'_i| + 1$. Then, s/he submits an aggregate query over S' and receives a noisy result. Based on the observed result, s/he calculates the posterior belief distribution over the all S'_is. The S'_i with the highest posterior probability is selected as the potential S'.

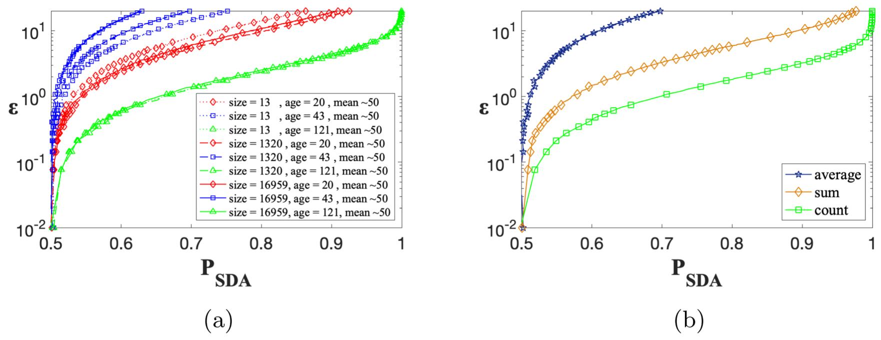

Fig. 2. (a) P_{SDA} for *average* query and different values of ε. (b) Alice's secret leakage in term of P_{SDA} by different queries and different values of ε.

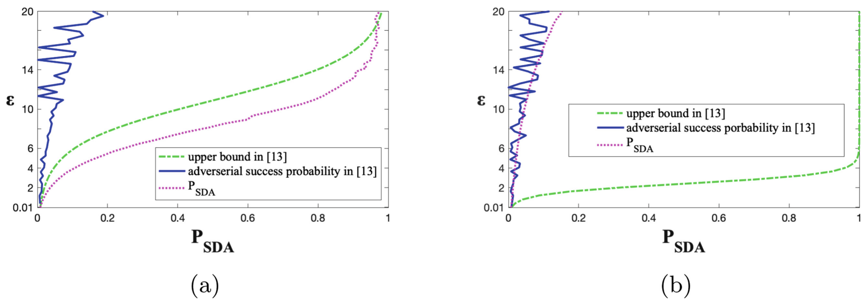

Fig. 3. Comparing two adversarial models for (a) *sum* and (b) *average* queries. (Color figure online)

Consequently, the missing person is revealed. [13] suggests ε shall be selected according to restricting the adversary's posterior belief when the missing person owns the maximum value among all the members.

The blue lines in Fig. 3 illustrate how the adversary's posterior belief of the S' depends on ε. The oscillations are due to the stochastic nature of the DP mechanism chosen by [13]. Resubmitting the query for the same value of ε results in a different posterior belief distribution. The green lines in Fig. 3 show the upper bound provided by [13]. The tightness of the upper bound depends to the distribution of data. Due to the different scenarios, we take few steps to properly compare P_{SDA} and [13]. First, we build a scenario where both methods can be used. Such a scenario is similar to the running example, with the additional constraint that Alice's secret is zero, i.e., she is the only female subscriber that did not watch *Channel*_1. We draw 120 random samples from the normal distribution $\mathcal{N}(74.5, 9)$, in the range $[0, 121]$ to serve as ages of individuals in S. The selected numbers and distribution are based on the real viewership of a randomly chosen channel for a randomly chosen period. We consider the maximum value from the sampled data as Alice's *age* since [13] suggests to set ε based on

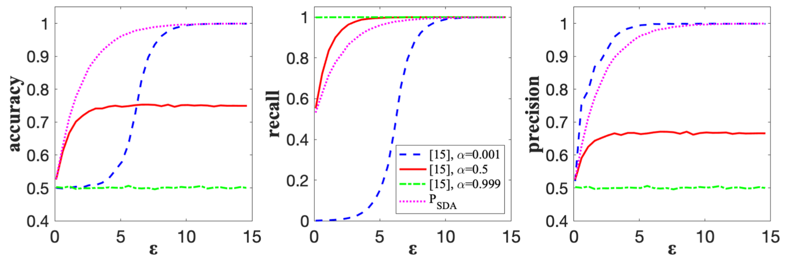

Fig. 4. Comparison with [15]: *accuracy*, *recall*, and *precision* for *count* queries.

protecting the secret of a participant with the maximum value. Second, we need incorporate the prior of [13] into the adversary model of P_{SDA}. P_{SDA} equals to $Pr(guess = secret)$ while the probability of adversarial success in [13] equals to $Pr(guess = secret|secret = 0)$ and $Pr(secret = 0) = \frac{1}{n}$, where n is the size of the dataset. One can calculate the former probability based on P_{SDA} by using a probability tree diagram. Third, the adversary in [13] knows the attributes of the users in the dataset, excluding the secret. This translates to the case where the actual value of Q_1^s or Q_1^a is known in the current study. Thus, both adversaries submit only one query to carry out the attack. As a consequence, we can construct an attack following our procedure for both *sum* and *average* queries based on the same assumptions as made by [13].

Figure 3a shows the comparison for the *sum* query. We plot both the posterior belief and the upper bound of [13] as described in their article. Our method outperforms [13] in gaining knowledge about the secret as it offers a higher probability of guessing the secret for any given ε. Note that in contrast to the previous plots, this plot starts close to 0, the prior probability is $\frac{1}{120} \approx 0.0083$.

For the *average* query, both adversaries perform similarly, as shown in Fig. 3b. Note that the shown upper bound does not indicate that the adversary of [13] outperforms the one of our model. [13] suggests using the adversary's posterior belief (blue lines in Fig. 3) to set ε when the upper bound is not tight, which occurs when the dataset does not contain outliers that significantly deviate from the average. The solution in [13] can not provide a value for ε in case of *count* queries as the size of S' is given.

Choosing ε based on the upper bound proposed in [13] may not protect against all adversarial models. As shown in Fig. 3a, our adversary may leak information from data protected by that mechanism. Also, the gaps between the blue and green lines in Fig. 3 indicate that the upper bound is not tight. Therefore, we may end up adding more noise than necessary.

Comparing P_{SDA} with Liu et al., 2019. Liu et al. [15] use hypothesis testing to guess the secret, i.e., a value that is either zero or one. To compare our method with [15], we implemented their attack model. Such a model requires the adversary to be aware of ε, ΔQ, Q_2, and a parameter $\alpha \in [0, 1]$. α controls the false alarm rate in their hypothesis testing. For αs close to one, the attack model in [15] is more prone to guess the secret to be one than zero and vice-versa.

Figure 4 plots the *accuracy*, *recall*, and *precision* of the attack models of [15] and our approach for three values of α. We run both attack models for 10000 times to guess the randomly generated secret for the *count* query. For a fair comparison, both adversaries submit only one query, as [15] assumes that the true value of Q_2 is public. As illustrated in Fig. 4, the two approaches perform similarly in detecting where $secret = 1$ for $\alpha = 0.5$ (they have similar *recall*). However, our attack model achieves higher *accuracy* and *precision* than [15] in detecting the two values of secret for $\alpha = 0.5$. Note that in [15], one can tune αs to either slightly outperform our approach in terms of recall ($\alpha = 0.999$)) or perform similarly in terms of precision ($\alpha = 0.0001$). In those cases, however, our approach outperforms [15]'s approach in the two other metrics. This may be a consequence of [15]'s goal of "only" protecting the cases where the secret is 1.

8 Discussion

In this section we discuss the application of P_{SDA}, we elaborate on other possible attacks, and future works.

Application of P_{SDA}**.** The notion presented in this paper can be used in many interactive data analysis settings that provide DP guarantees [1,5,8,9,16,18,19]. These settings are mostly offered as a library or programming platform that facilitates data curators to build a private interface for interacting with the data without privacy violations (e.g., due to the wrong implementations). However, to benefit from the privacy preservation of these packages, the data curator still needs to determine a good value for ε. The discussion here relies on PINQ [16] a LINQ based platform designed for differentially private interactive analysis of a dataset, to exemplify the challenges arising from this choice and how our contribution can help address these.

PINQ supports major aggregation queries like *count*, *sum*, *average*, and *median*. It also contains differentially private implementations of operations like *select*, *where*, and *groupby*. The count and sum queries used in this paper follow the same algorithms. For average queries we use the algorithm from [14], as [14] shows that the PINQ *average* operator does not satisfy ε-differential privacy.

One of the access policy provided by PINQ is fixed budget access, which allows the data curator to decide on the value of ε as a fixed budget for the user. Then it is up to the user to distribute this budget among the aggregations and operations of his/her interest. PINQ follows the composition theorem and it does not allow the analyst to run a query with ε exceeding the allocated budget. In terms of P_{SDA}, a fixed budget setup of PINQ is as secure as its weakest aggregation, which, looking at the plots in Fig. 2b, is the *count* queries. As a result, the maximum budget should be allocated by focusing on the count query.

Other Threats. We discuss two other threats from the scenario in Sect. 2. We did not consider *differencing attacks with more than two queries*, as we focused on attacks aiming at isolating the target person with one or two queries. The cases with one or two queries are the ones where differencing attacks are the most successful. According to the composition theorem (Sect. 4), asking more queries results in noisier answers, as the available budget is shared among the queries. Also, the linear combination of two Laplace distributions (both centered at the same mean value) is flatter than the two distributions themselves [17]. As the number of distributions increases, the result becomes flatter and closer to the uniform distribution. This makes it hard to make a decision with higher success. Thus, if an adversary can isolate a person with multiple queries, it is better to merge these queries to only one query (or two queries, in case the dataset may have a restriction on querying minorities). Even though this query may require a complex condition, it has no effect on the privacy budget consumption [16]. As a consequence, setting ε based on differencing attacks with one or two queries protects also against differencing attacks with three or more queries.

Selecting ε with our proposed method is enough to protect the dataset against *linear reconstruction attacks*. A linear reconstruction attack is an attack for reconstructing a private dataset (fully or partially) by running several aggregate queries on the dataset. According to [3], a linear reconstruction attack's success depends on three parameters: 1) the number of participants in the dataset n, 2) the number of released noisy aggregates m, and 3) the error bound added to the aggregate queries E. This attack works in two settings: in one setting, m is exponential in n, and the error E is linear in n. In the second setting, m is polynomial in n, and the error E is on the order of $\sqrt{n}$.

In the DP setting, according to the composition theorem, the noise order depends only on m, and not n. If we equally distribute the budget among the queries, m and E are linearly related. For a fixed and sufficiently large n and a given ε, the adversary cannot achieve a successful reconstruction attack. As explained in Sect. 2 Footnote (See Footnote 1), we assume that the protected dataset does not allow querying over a subset smaller than some threshold. Hence, provided that such a threshold is large enough to mitigate a reconstruction attack, the differencing attack is a more relevant threat than the reconstruction attack.

Future Work. In the DP literature there is a huge misalignment between methods to set DP parameters and design of novel mechanisms and their application in different use cases [21]. To fill this gap, we started analysing base mechanisms, such as *count*, *sum*, and *average*. The next step of our research is to extend our framework to other widely used mechanisms such as median or histograms. Our final goal is to develop a general framework that covers a large amount of queries.

For the mechanisms which employ distributions beyond Laplace distribution, the introduced attack strategy in Sect. 5.2 can be adapted to any symmetric distribution for which the formula for a linear combination is calculable. Exploring the effect of different distributions on P_{SDA} is another direction for future work.

To mitigate the adversarial threat introduced in this paper, one possible solution is to add fake users with similar features to the users vulnerable to differencing attacks using techniques such as k-anonymity [12]. Investigating how such a method will impact P_{SDA} can be another direction to extend the current study.

9 Conclusions

For DP to be trusted, we need to investigate transparent approaches for setting ε. The P_{SDA} (the probability of successful differencing attack) approach presented in this paper—whilst limited to three query types and exhibiting certain underlying assumptions—presents an important step in this direction.

Our findings show that, for a given value for ε, the probability of a privacy leak varies depending on the query type and the target person's data. Furthermore, comparison of our method with the methods of [13,15] shows that the privacy loss of a given value of ε differs for different adversarial models. Our proposed adversary carries out equal or more successful attacks than the adversaries proposed in [13] and [15]. Moreover, our method is designed to work in more generic scenarios than the ones considered by [13]. Hence, to the best of our knowledge, our P_{SDA} provides a more general approach for choosing an appropriate ε than either of these related works.

Acknowledgements. We thank the Swiss National Science Foundation for their partial support under contract number #407550_167177.

References

1. Beck, M., Bhatotia, P., Chen, R., Fetzer, C., Strufe, T., et al.: PrivApprox: privacy-preserving stream analytics. In: 2017 USENIX Annual Technical Conference USENIX ATC 2017, pp. 659–672 (2017)
2. Clopper, C.J., Pearson, E.S.: The use of confidence or fiducial limits illustrated in the case of the binomial. Biometrika **26**(4), 404–413 (1934)
3. Dinur, I., Nissim, K.: Revealing information while preserving privacy. In: Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 202–210 (2003)
4. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**(3–4), 211–407 (2014)
5. Gaboardi, M., et al.: Psi (ψ): a private data sharing interface. arXiv preprint arXiv:1609.04340 (2016)
6. Garfinkel, S.L., Abowd, J.M., Powazek, S.: Issues encountered deploying differential privacy. In: Proceedings of the 2018 Workshop on Privacy in the Electronic Society, pp. 133–137 (2018)
7. Hsu, J., et al.: Differential privacy: an economic method for choosing epsilon. In: 2014 IEEE 27th Computer Security Foundations Symposium, pp. 398–410. IEEE (2014)
8. Johnson, N., Near, J.P., Hellerstein, J.M., Song, D.: Chorus: a programming framework for building scalable differential privacy mechanisms. In: 2020 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 535–551. IEEE (2020)
9. Johnson, N., Near, J.P., Song, D.: Towards practical differential privacy for SQL queries. Proc. VLDB Endow. **11**(5), 526–539 (2018)
10. Kotz, S., Kozubowski, T., Podgorski, K.: The Laplace Distribution and Generalizations: A Revisit with Applications to Communications, Economics, Engineering, and Finance. Springer, Heidelberg (2012). https://doi.org/10.1007/978-1-4612-0173-1

11. Krehbiel, S.: Choosing epsilon for privacy as a service. Proc. Priv. Enhanc. Technol. **2019**(1), 192–205 (2019)
12. Latanya, S.: k-anonymity: A model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **10**(05), 557–570 (2002)
13. Lee, J., Clifton, C.: How much is enough? Choosing ε for differential privacy. In: Lai, X., Zhou, J., Li, H. (eds.) ISC 2011. LNCS, vol. 7001, pp. 325–340. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24861-0_22
14. Li, N., Lyu, M., Su, D., Yang, W.: Differential privacy: from theory to practice. Synth. Lect. Inf. Secur. Priv. Trust **8**(4), 1–138 (2016)
15. Liu, C., He, X., Chanyaswad, T., Wang, S., Mittal, P.: Investigating statistical privacy frameworks from the perspective of hypothesis testing. Proc. Priv. Enhanc. Technol. **2019**(3), 233–254 (2019)
16. McSherry, F.D.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 19–30. ACM (2009)
17. Nadarajah, S.: The linear combination, product and ratio of Laplace random variables. Statistics **41**(6), 535–545 (2007)
18. Proserpio, D., Goldberg, S., McSherry, F.: Calibrating data to sensitivity in private data analysis: a platform for differentially-private analysis of weighted datasets. Proc. VLDB Endow. **7**(8), 637–648 (2014)
19. Roy, I., Setty, S.T., Kilzer, A., Shmatikov, V., Witchel, E.: Airavat: security and privacy for mapreduce. In: NSDI, vol. 10, pp. 297–312 (2010)
20. Tao, Y., He, X., Machanavajjhala, A., Roy, S.: Computing local sensitivities of counting queries with joins. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 479–494 (2020)
21. Wagner, I., Eckhoff, D.: Technical privacy metrics: a systematic survey. ACM Comput. Surv. (CSUR) **51**(3), 1–38 (2018)
22. Wood, A., et al.: Differential privacy: a primer for a non-technical audience. Vand. J. Ent. Tech. L. **21**, 209 (2018)