View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by INRIA a CCSD electronic archive server

# Preserving differential privacy under finite-precision semantics

Ivan Gazeau, Dale Miller, Catuscia Palamidessi

## To cite this version:

Ivan Gazeau, Dale Miller, Catuscia Palamidessi. Preserving differential privacy under finite-precision semantics. Theoretical Computer Science, Elsevier, 2016. hal-01390927

**HAL Id: hal-01390927**

**https://hal.inria.fr/hal-01390927**

Submitted on 2 Nov 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Preserving differential privacy under finite-precision semantics

Ivan Gazeau[1], Dale Miller[2], and Catuscia Palamidessi[2]
[1]University of Birmingham,[2] INRIA and LIX, Ecole Polytechnique

**Abstract**

The approximation introduced by finite-precision representation of continuous data can induce arbitrarily large information leaks even when the computation using exact semantics is secure. Such leakage can thus undermine design efforts aimed at protecting sensitive information. We focus here on differential privacy, an approach to privacy that emerged from the area of statistical databases and is now widely applied also in other domains. In this approach, privacy is protected by adding noise to the values correlated to the private data. The typical mechanisms used to achieve differential privacy have been proved correct in the ideal case in which computations are made using infinite-precision semantics. In this paper, we analyze the situation at the implementation level, where the semantics is necessarily limited by finite precision, i.e., the representation of real numbers and the operations on them are rounded according to some level of precision. We show that in general there are violations of the differential privacy property, and we study the conditions under which we can still guarantee a limited (but, arguably, acceptable) variant of the property, under only a minor degradation of the privacy level. Finally, we illustrate our results on two examples: the standard Laplacian mechanism commonly used in differential privacy, and a bivariate version of it recently introduced in the setting of privacy-aware geolocation.

**Keywords:** Differential privacy, floating-point arithmetic, robustness to errors.

## 1. Introduction

It is well known that, due to the physical limitations of actual machines, in particular the finiteness of their memory, real numbers and their operations cannot be implemented with full precision. While for traditional computation getting an approximate result is not critical when a bound on the error is known, we argue that, in security and privacy applications, the approximation error can became a fingerprint potentially causing the disclosure of secret or confidential information.

We investigate here the confidentiality issues caused by the use of finite precision. More specifically, we study the information leaked about the input data that can be caused by the errors in the result, assuming that the adversary knows the result and the way the program works (because the program may be public, for example). The standard techniques to measure security breaches do not apply because those techniques

analyze of the *ideal* system—i.e. using *exact*) semantics—and do not reveal the information leaks caused by the implementation.

Consider, for instance, the following simple program

$$\text{if } \ f(h) > 0 \ \text{ then } \ \ell = 0 \ \text{ else } \ \ell = 1,$$

where $h$ is a high (i.e., confidential) variable and $\ell$ is a low (i.e., public) variable. Assume that $h$ can take two values, $v_1$ and $v_2$, and that both $f(v_1)$ and $f(v_2)$ are strictly positive. Then, in the ideal semantics, the program is perfectly secure, i.e. it does not leak any information. However, in the implementation, it could be the case that the test succeeds in the case of $v_1$ but not in the case of $v_2$ because, for instance, the value of $f(v_2)$ is below the smallest representable positive number. Hence, we would have a total disclosure of the secret value.

While the example above is elementary, it illustrates the pervasive nature of this problem and the impact it can have on confidentiality. Clearly, the problem of working without full precision should receive adequate treatment.

In this paper, we consider a more tricky case where the agent returns a noisy answer by using random numbers. Often the addition of random noise is done on purpose to prevent access to secret values. We will see that, however, since noises are generated in finite precision, even noise contains computational errors that allows an attacker to retrieve secrets.

In order to have concretes examples and to analyze existing specifications, we will study here the particular case of using finite precision with *differential privacy*. Differential privacy is an approach to the protection of private information in the field of statistical databases. Statistical databases are databases containing individual records, and aim at supporting the discovery of aggregate information, such as plausible causes for diseases, social normal, or trends public opinion. Of course, statistical databases have to preserve the anonymity and privacy of participants: It is likely that people will not participate to a survey if they know that their personal information will be revealed to anybody. However, as we explain in section 2.3, granting anonymity and privacy is not a trivial task.

Differential privacy has been first proposed in [6, 7] as a formal approach to preserve the anonymity and privacy of the participants in a statistical database. This approach is now being used in many other domains ranging from programming languages [2, 8] to social networks [16] and geolocation [14, 11, 1].

The key idea behind differential privacy is that whenever someone queries a dataset, the reported answer should not allow him to distinguish whether a certain individual record is in the dataset or not. More precisely, the presence or absence of the record should not change significantly the probability of obtaining a given answer. The standard way of achieving such a property is by using an *oblivious mechanism*[1] which consists in adding some noise to the true answer. Now the point is that, even if such a mechanism is proved to provide the desired property in the ideal semantics, its implementation may induce errors that alter the least significant digits of the reported answer

[1]The name "oblivious" comes from the fact that the final answer depends only on the answer to the query and not on the dataset.

and cause significant privacy breaches. Let us illustrate the problem with an example.

**Example 1.1.** Consider the simplest representation of reals: fixed-point numbers. This representation is used on low-cost processors which typically do not have a floating-point arithmetic module. Each value is stored in a memory cell of fixed length. In such cells, the last $d$ digits represent the fractional part. Thus, if the value (interpreted as an integer) stored in the cell is $z$, its semantics (i.e., the true real number being represented) is $z \cdot 2^{-d}$.

To grant differential privacy, the standard technique consists in returning a random value with probability $p(x) = {}^1\!/_{2b} \cdot \exp(-{}^{|x-r|}\!/_b)$ where $r$ is the true result and $b$ is a scale parameter which depends on the degree of privacy to be obtained and on the sensitivity of the query. The sensitivity of the query is the maximal difference in the result when one entry is removed or added. To get a random variable with any specific distribution, in general, we need to start with an initial random variable provided by a primitive of the machine with a given distribution. To simplify the example, we assume that the machine already provides a Laplacian random variable $X$ with a scale parameter 1. The probability distribution of such an $X$ is $p_X(x) = {}^1\!/_2 \exp(-|x|)$. Hence, if we want to generate the random variable $bX$ with probability distribution

$$p_{bX}(x) = {}^1\!/_{2b} \cdot \exp(-{}^{|x|}\!/_b),$$

we can just multiply by $b$ the value $x = z \cdot 2^{-d}$ returned by the primitive.

Assume that we want to add noise with a scale parameter $b = 2^n$ for some fixed integer $n$ ($b$ can be big when the sensitivity of the query and the required privacy degree are high). In this case, the multiplication by $2^n$ returns a number $2^n z \cdot 2^{-d}$ that, in the fixed-point representation, terminates with $n$ zeroes. Hence, when we add this noise to the true result, we return a value whose representation has the same $n$ last digits as the secret. For example, assume $b = 2^2 = 4$ and $d = 6$. Consider that the true answers are $r_1 = 0$ and $r_2 = 1 + 2^{-5}$. In the fixed-point representation, the last two digits of $r_1$ are 00, and the last two digits of $r_2$ are 10. Hence, even after we add the noise, it is still possible to determine which was the true value between $r_1$ or $r_2$. Note that the same example holds for every $b = 2^n$ and every pair of true values $r_1$ and $r_2$ which differ by ${}^{(2^n k+h)}\!/_{2^d}$ where $k$ is any integer and $h$ is any integer between 1 and $2^n - 1$. Figure 1 illustrates the situation for $n = 2$, $b = 4$, $d = 6$, $k = 3$ and $h = 2$. End of Example 1.1.

Another attack, based on the IEEE standard floating-point representation [12], was presented in [15]. In contrast to [15], we have chosen an example based on the fixed point representation because it allows to illustrate more distinctively a problem for privacy which rises from the finite precision[2] and which is, therefore, pandemic. This is not the case for the example in [15]: fixed-point and integer-valued algorithms are immune to that attack.

In this paper, we propose a solution to fix the privacy breach induced by the finite-precision implementation of a differentially-private mechanism for any kind of implementation. Our main concern is to establish a bound on the degradation of privacy

[2]More precisely, the problem is caused by scaling a finite set of randomly generated numbers. It is easy to prove that the problem raises for any implementation of numbers, although it may not raise *for every point* like in the case of the fixed-point representation.

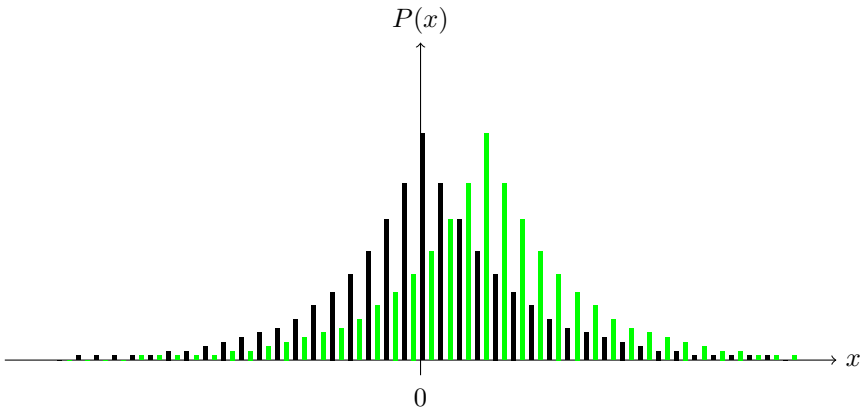


Figure 1: The probability distribution of the reported answers after the addition of Laplacian noise $P_{bX}(x)$ with $b = 4$, for the true answer $r_1 = 0$ (black) and $r_2 = 3 \cdot 2^{-4} + 2^{-5}$ (green).

induced by both the finite representation and by the computational errors in the generation of the noise. In order to achieve this goal, we use the concept of *closeness* which allows us to reason about the approximation errors and their accumulation. In addition, we make as few assumptions as possible about the procedure for generating the noise. In particular, we do not assume that the noise has a linear Laplacian distribution: it can be any noise that provides differential privacy and whose implementation satisfies a few properties (normally granted by the implementation of real numbers) which ensure its closeness. We illustrate our method with two examples: the classic case of the univariate (i.e., linear) Laplacian, and the case of the bivariate Laplacian. The latter distribution is used, for instance, to generate noise in privacy-aware geolocation mechanisms [1].

### *1.1. Related work*

Mironov has discovered independently the problem introduced by the finite precision in the implementation of differential privacy [15]. As already mentioned, that paper showed an attack on the Laplacian-based implementation of differential privacy within the IEEE standard floating-point representation[3]. To thwart such an attack, the author of [15] proposed a method that avoids using the standard uniform random generator for floating point (because it does not draw all representable numbers but only multiple of $2^{-52}$). Instead, his method generates two integers, one for the mantissa and one for the exponent in such a way that every representable number is drawn with its correct probability. Then it computes the linear Laplacian using a logarithm implementation (assumed to be full-precision), and finally it uses a snapping mechanism consisting in truncating large values and then rounding the final result.

The novelties of our work, w.r.t. [15], consist in the fact that we deal with a general kind of noise, not necessarily the linear Laplacian, and with any kind of implementa-

[3]We discovered our attack independently, but [15] was published first.

tion of real numbers, not necessarily the IEEE floating point standard. Furthermore, our kind of analysis allows us to measure how safe an existing solution can be and what to do if the requirements needed for the safety of this solution are not met. Finally, we provide a correct implementation of the bivariate Laplacian also as a valuable contribution, given its practical usefulness for location-based applications.

The only other work we are aware of that considers both computational error and differential privacy is [4]. However, that paper does not consider at all the problem of the loss of privacy due to implementation error: rather, they develop a technique to establish a bound on the error, and show that this technique can also be used to compute the sensitivity of a query, which is a parameter of the Laplacian noise.

### *1.2. Plan of the paper*

This paper is organized as follow. In section 2, we recall some mathematical definitions as well as the basis of differential privacy and we introduce some notation. In section 3, we discuss how finite precision causes errors in the result of the algorithm. We modify the initial algorithm in 3.5 to deal with some of these errors. In section 4, we use the bounds about the error found in the previous section to get a bound on the increase of the $\epsilon$ parameter due to the finite implementation. The two sections that follow present some applications of our result: Section 5 illustrates the technique for the case of Laplacian noise in one dimension and section 6 shows how our theorem applies to the case of the Euclidean bivariate Laplacian. Section 7 concludes.

## 2. Preliminaries and notation

In this section, we recall some definitions and we introduce some notation that we use throughout the paper.

### *2.1. Geometrical notations*

There are several natural definitions of distance on $\mathbb{R}^m$ [17]. For $m \in \mathbb{N}$ and $x = (x_1, \ldots, x_m) \in \mathbb{R}^m$, the $L_p$ norm of $x$, which we will denote by $\|x\|_p$, is defined as $\|x\|_p = \sqrt[p]{\sum_{i=1}^m |x_i|^p}$. The corresponding distance function is $d_p(x, y) = \|x - y\|_p$. We extend this norm and distance to $p = \infty$ in the usual way: $\|x\|_\infty = \max_{i \in \{1,\ldots,m\}} |x_i|$ and $d_\infty(x, y) = \|x - y\|_\infty$. The notion of $L_\infty$ norm is extended to functions in the following way: given $f : A \to \mathbb{R}^m$, we define $\|f\|_{p,\infty} = \max_{x \in A} \|f(x)\|_p$. When clear from the context, we will omit the parameter $p$ and write simply $\|x\|$, $d(x, y)$ and $\|f\|_\infty$ for $\|x\|_p$, $d_p(x, y)$ and $\|f\|_{p,\infty}$ respectively.

Let $S \subseteq \mathbb{R}^m$. The *diameter of* $S$ is defined as $\varnothing(S) = \max_{x,y \in S} d(x, y)$. For $x \in \mathbb{R}^m$, the *translations* of $S$ by $x$ and $-x$ are defined as $S + x = \{y + x \mid y \in S\}$ and $S - x = \{y - x \mid y \in S\}$ where $+$ and $-$ are the vectorial spaces operators.

**Definition 2.1.** *We define the expansion of the set $S$ by $r \in \mathbb{R}^+$ :* $\mathrm{expand}(S, +r) = \{x \mid \exists s \in S, d(x, s) \leq r\}$ *and the shrinking of the set $S$ by $r \in \mathbb{R}^+$:* $\mathrm{expand}(S, -r) = \{x \mid \forall s \in \mathbb{R}^m, d(x, s) \leq r \implies s \in S\}$.

Note that these two definitions are related as follows: $\mathbb{R}^m \setminus \mathrm{expand}(S, -\epsilon) = \mathrm{expand}(\mathbb{R}^m \setminus S, +\epsilon)$.

*2.2. Probability and measure theory*

In this section, we recall the basic notions of measure theory.

**Definition 2.2** ($\sigma$-algebra and measurable space)**.** *A $\sigma$-algebra $\mathcal{T}$ for a set $M$ is a nonempty set of subsets of $M$ that is closed under complementation (wrt to $M$) and (potentially empty) enumerable union. The tuple $(M, \mathcal{T})$ is called a* measurable space*.*

On $\mathbb{R}^m$, we denote by $\mathcal{S}$ the Lebesgue's $\sigma$-algebra that contains all hypercube.

**Definition 2.3** (Probability space)**.** *A probability space is a tuple $(\Omega, \mathcal{F}, P)$ where $\Omega$ is a sample space (i.e., the set of all possible outcomes), $\mathcal{F}$ is a set of events (where each event is a set of zero or more outcomes) which is also a $\sigma$-algebra on $\Omega$, and $P$ is a probability measure on $(\Omega, \mathcal{F})$ (i.e., a measure such that $P(\Omega) = 1$).*

**Definition 2.4** (Random variable)**.** *Let $(\Omega, \mathcal{F}, P)$ be a probability space and $(E, \mathcal{E})$ be a measurable space. Then a random variable is a measurable function $X : \Omega \to E$. We shall use the expression $P[X \in B]$ to denote $P\left(X^{-1}(B)\right)$.*

**Definition 2.5** (Support)**.** *The support of a measure is the set of all $x$ where the measure of some neighborhood of $x$ is not null. By extension, the support of a random variable $X$ is the support of its corresponding measure.*

We will make use of the Lebesgue measure $\lambda$ on $(\mathbb{R}^m, \mathcal{S})$ where $\mathcal{S}$ is the Lebesgue $\sigma$-algebra. This is the standard way of assigning a measure to subsets of $\mathbb{R}^m$.

In this paper, we use the following general definition of the Laplace distribution (centered at zero).

**Definition 2.6** (Laplace distribution)**.** *The density function $F$ of a Laplace distribution with scale parameter $b$ is $F_b(x) = K(b)\exp(-b\|x\|)$ where $K(b)$ is a normalization factor which is determined by imposing $\int_S F_b(x)\,dx = 1$.*

*2.3. Differential privacy*

In this section, we recall the definition of differential privacy on databases. We will denote by $\mathcal{D}$ the set of all possible databases. We start by formalizing the notion of presence versus non-presence of an individual.

**Definition 2.7** (adjacent databases)**.** *Given two databases $D_1$ and $D_2$ in $\mathcal{D}$, we denote by $D_1 \sim D_2$ the fact that $D_1$ and $D_2$ differ by exactly one row. Namely, $D_2$ is obtained from $D_1$ by adding or removing the data of one individual.*

In order to protect the private data of an individual, the differential privacy framework uses randomization to obfuscate the answers to queries.

**Definition 2.8** (randomized mechanism)**.** *A randomized mechanism $\mathcal{K}$ is a function that takes a database $D$ in $\mathcal{D}$ and a query $q$ and returns a random variable $X$: $X = \mathcal{K}_q(D)$. We omit the $q$ parameter when there is no ambiguity.*

We can now provide the formal definition of differential privacy. The idea is that the probability of obtaining a certain answer to the query is almost the same whether an individual is present or not, hence the answer does not provide too much probabilistic information about the data of the individual.

**Definition 2.9** ($\epsilon$-differential privacy)**.** *A randomized mechanism $\mathcal{K} : \mathcal{D} \to \mathbb{R}^m$ is $\epsilon$-differentially private if for all databases $D_1$ and $D_2$ in $\mathcal{D}$ with $D_1 \sim D_2$, and all $S \in \mathcal{S}$ (the Lebesgue $\sigma$-algebra on $\mathbb{R}^m$), we have :*

$$P[\mathcal{K}(D_1) \in S] \leq \exp(\epsilon) P[\mathcal{K}(D_2) \in S]$$

*2.4. Standard technique to implement differential privacy*

We will focus on the so-called oblivious mechanisms, which are the most used, and are defined as follow.

**Definition 2.10** (oblivious mechanisms)**.** *A mechanism is oblivious if it is obtained by applying a random function to the true answer to the query, i.e., not directly to the database. We use the notation $f_q(D)$ to denote the true answer on the database $D$ and $f(D)$ when there is no ambiguity about $q$.*

To calibrate the noise of an oblivious mechanism we do not need to consider the databases anymore, the only interesting parameter is the maximal deviation that one individual can generate for a given query. This leads to the following definition.

**Definition 2.11** (sensitivity)**.** *The sensitivity $\Delta_f$ of a function $f : \mathcal{D} \to \mathbb{R}^m$ is*

$$\Delta_f = \sup_{\substack{D_1,\, D_2 \,\in\, \mathcal{D} \\ D_1 \,\sim\, D_2}} d(f(D_1), f(D_2)).$$

We are interested in the particular case of oblivious mechanisms that only add a random value to the result. These are the most common oblivious mechanisms and we will limit our study to them.

**Mechanism 1.**

$$\mathcal{K}(D) = f(D) + X \tag{1}$$

Another important feature of a mechanism is its *utility*. For instance, a function that only returns a random variable is 0-differential private but since it never allows to learn anything, as a mechanism it would be perfectly useless. In the case of an additive mechanism the utility can be defined as a function of the variance between the true answer and the real value.

A mechanism that offers a good trade-off between privacy and utility is the so-called Laplacian mechanism, which is defined as in (1), where $X$ is a Laplacian calibrated by $\epsilon$ and the sensitivity of the query. Namely, the density function of $X$ is $F_b(x)$ (cf. Definition 2.6) with $b = {}^{\Delta_{f_q}}/_{\epsilon}$.

## 3. Measuring the computational error

This section aims at measuring and bounding the computational errors that the finite semantics induces in the implementation of the mechanism. First, we discuss the influence of the error in the function computing the true answer. Then, we analyze the deviation between a theoretical perfect random generator and a pseudo random generator in finite precision. Finally, we show that the straightforward implementation is not safe and we provide a solution to fix that problem.

### 3.1. Approximate computation of the true answer to the query

Let $f$ be the function computing the true answer of the query, and $f'$ its implementation in finite precision.

The difference between $f'$ and $f$ may affect the utility of the answer, but in general it does not influence the level of differential privacy. In particular, in the case of an oblivious mechanism, the way the deviation on $f$ influences the level of privacy is only via the features of $f$ that are used for the computation of the noise. Typically such feature is the sensitivity of the query, $\Delta_f$. For example, the definition of Laplacian noise depends on the parameter $b$ which is $\Delta_f$.

### 3.2. Implementation of real valued random variables

Computer systems usually provide a primitive that implements a random variable $U$ of range $[0, 1[$ with uniform distribution[4]. Other random variables and their density functions can usually be defined as functions of this $U$. For instance $X = n(U) = 4U - 2$ is a uniform random variable of range $[-2, 2[$. For random variable whose support (co-domain) is in $\mathbb{R}^n$, the definition can make use of a tuple $U$ of several independent uniform random variables $U_1, \ldots, U_m$. For instance, $(X, Y) = n(U_1, U_2) = (U_2 \cos(2\pi U_1), U_2 \sin(2\pi U_1))$ is a random variable whose support is the unit disc.

In this paper, we assume that the implementation of the noise in Mechanism 1 is based on such $U$ and $n$. Hence, we only consider mechanisms in which the random function $X$ of (1) is of the form $n(U)$:

**Mechanism 2.**

$$\mathcal{K}_n(D) = f(D) + n(U) \tag{2}$$

In the following we consider the errors introduced by the emulations of those $U$ and $n$ in finite precision.

### 3.3. Error due to the initial random generator

A perfect uniform random generator would generate any value in $[0, 1[$ with uniform distribution. However, since the implementation can only returns values with a finite precision, the distribution is not uniform. Here we study how this deviation influences the error in the mechanism.

We denote by $U$ the tuple of $m$ uniform random variables used to generate the noise in the exact semantics. In other words, $U$ is a uniform random variable in $[0, 1[^m$. We denote by $U'$ the pseudo random variable in $[0, 1[^m$ generated in the finite precision semantics. To model the error, we consider $U' = n_0(U)$ where $n_0$ is a measurable function of type $[0, 1[^m \to [0, 1[^m$ that maps any $u \in [0, 1[^m$ to the finite set of possible outputs of our pseudo random generator. In summary, $U, U'$ are random variables of type $\Omega \to [0, 1[^m$, and $U'(\omega) = n_0(U(\omega))$.

[4]In general, a machine provides a pseudo-random generator $U$, not a "real" random one. However the question of the reliability of pseudo randomness is out of the scope of this paper, so we will assume that $U$ is really random.

For instance, if our precision allows to represent only numbers that are multiples of $2^{-53}$, then we can model $U'$ with the function $n_0$ that takes a real number and truncates it after the $53^{\text{th}}$ decimal bit.

We will use $\delta_0$ to denote the error due to this initial random generator:

**Definition 3.1.** *The initial error $\delta_0 \in \mathbb{R}^+$ is the maximal distance between $n_0(u)$ and $u$ for any $u \in [0,1[^m$ :*

$$\delta_0 = \|n_0 - Id\|_\infty.$$

*where* Id *is the identity function.*

For instance, in case $U'$ generates numbers that are multiple of $\delta$, we have $\delta_0 = \delta$. In our previous example we have $\delta = 2^{53}$.

*3.4. Errors due to the transformation function*

We now consider the error in the implementation of the function $n$ which is used to transform the initial uniform $U$ into a noise $n(U)$ with the desired distribution. We denote by $n'$ the actual function resulting by implementing $n$ in finite-precision. and by $X' = n'(U') = n'(n_0(U))$ the random variable actually generated.

In order to bound the maximal error between the result generated by the implementation and the exact one, we use the notion of closeness that was defined in [9].

**Definition 3.2** ($(k,\delta)$-closedness, [9])**.** *Let $A$ and $B$ be metric spaces with distance $d_A$ and $d_B$, respectively. Let $n$ and $n'$ be two functions from $A$ to $B$ and let $k, \delta \in \mathbb{R}^+$. We say that $n'$ is $(k,\delta)$-close to $n$ if*

$$\forall u, v \in A, d_B(n(u), n'(v)) \leq k\ d_A(u,v) + \delta.$$

The closeness property is related to the one of being $k$-Lipschitz, as shown in [9]:

**Proposition 3.1** ([9])**.** *If $n$ is $k$-Lipschitz and $\|n - n'\|_\infty \leq \delta$ then $n$ and $n'$ are $(k,\delta)$-close.*

**Corollary 3.1.** *$n_0$ and Id are $(1,\delta_0)$-close.*

**Proof** From Definition 3.1, the fact Id is 1-Lipschitz, and Proposition 3.1. □

If also $n$ and $n'$ were $(k,\delta)$-close for some $k$, then, by compositionality we could derive that $n$ and $n' \circ n_0$ are $(k, k\delta_0 + \delta)$-close as well, and this would allow to establish a bound on the divergence between $\mathcal{K}$ and its implementation $\mathcal{K}'$.

Unfortunately, $n$ and $n'$ are not $(k,\delta)$-close. To show this, we start with the following lemma which states that in a differentially private mechanism, if the noise is additive, then its amplitude is unbounded.

**Lemma 3.1.** *If a mechanism of the form* (1) *is differentially private, and there exist at least two adjacent databases where the query has two different answers, then the support of $X$ has infinite diameter.*

**Proof** Let $r_1$ and $r_2$ different answers on two adjacent databases. Let $M \in \mathcal{S}$ (the set of measurable sets) such that $\|M\|$ is bounded and $P(X \in M) > 0$. By definition of translations (cfr. Section 2.1), we also have $P(r_1 + X \in M + r_1) > 0$. Since $\mathcal{K}$ is $\epsilon$-differentially private, we must have $P(r_2 + X \in M + r_1) > 0$ as well, otherwise the ratio between the two probabilities would be infinite. Hence we have shown that $P(X \in M) > 0 \implies P(X \in M + r_1 - r_2) > 0$. Since this is valid for any $M$, we have in particular that for any $h \in \mathbb{N}$, $P(X \in M + h(r_1 - r_2)) > 0 \implies P(X \in M + (h+1)(r_1 - r_2)) > 0$. From the assumption $P(X \in M) > 0$ and the last implication, we conclude, by induction, that for all $h \in \mathbb{N}, P(X \in M + h(r_1 - r_2)) > 0$.

Finally, since $M$ is bounded, for any $r \in \mathbb{R}$ there exists a $h$ such that

$$\{x \in \mathbb{R}^m \mid d(0, x) \leq r\} \cap M + h(r_1 - r_2) = \emptyset \tag{3}$$

where 0 is the origin. (Note that $\{x \in [0, 1[^m \mid d(0, x) \leq r\}$ is the ball centered in 0 of radius $r$.) This means that the set $M + h(r_1 - r_2)$ does not have any element $x$ with $d(0, x) \leq r$. In summary, we have exhibited sets $(M + h(r_1 - r_2))$ with non null probability which are arbitrarily far from the origin. We can conclude that, for all $s \in \mathbb{R}$, $P(X > s) > 0$. □

Finally, we show that indeed the deviation caused by the finite precision, in the implementation of the noise, is unbounded:

**Theorem 3.1.** *If a mechanism of the form* (2) *is differentially private, then there is no* $\delta \in \mathbb{R}$ *such that*

$$\forall u \in [0, 1[^m,\ d((n' \circ n_0)(u), n(u)) \leq \delta$$

**Proof** Since there is only a finite set of possible values $n_0(u)$ for $u \in [0, 1[^m$, there exists a maximum value $x_M = \max\{\|n'(n_0(u))\| \mid u \in [0, 1[^m\}$. From Lemma 3.1, we have that, for every $\delta \in \mathbb{R}$, there exists $u \in [0, 1[^m$ such that $\|n(u)\| > \delta + x_M$, and therefore $d(n(u), n'(n_0(u))) > k$. □

In Figure 2, we illustrate the problem pointed out by Theorem 3.1, and the consequent impossibility for the implementation to achieve differential privacy on the whole domain of a mechanism of the form (2). We have considered a noise $n$ defined so that the ideal mechanism would be 0.2-differentially private, and we have simulated the distribution of the corresponding noise generated in fixed precision (where errors has been overplayed). The black and the green curves represent the right branch of the distributions $f(D_1) + n'(n_0(U))$ and $f(D_2) + n'(n_0(U))$ respectively, where $f(D_1) = 0$ and $f(D_2) = 1$. They are step functions: each step represents the probability for a given representable value. As we can see, while the ratio between the two distributions is (almost) $\exp(0.25)$ around 0, when the mechanism returns a larger noise the ratio becomes much higher, and for the value $x_M$ defined in the proof of Theorem 3.1 we have $P(f(D_1) + n'(n_0(U)) > x_M) = 0$ while $P(f(D_2) + n'(n_0(U)) > x_M) \neq 0$, which means that the ratio is infinite.

### *3.5. Truncating the result*

In this section we discuss how to modify the mechanism so to be able to bound the difference between $n$ and $n' \circ n_0$. One way of doing this is by truncating the result.

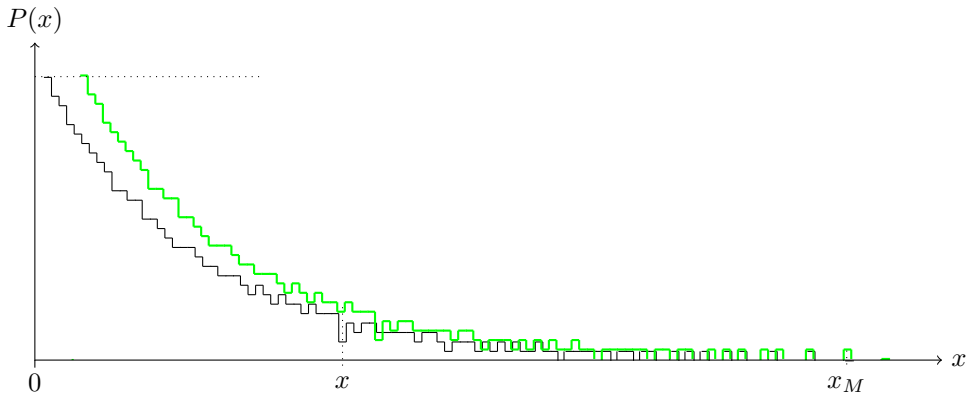


Figure 2: The effect of the finiteness of $n_0(U)$ on the implementation of the noise

Some differentially private mechanisms already use a truncation procedure. The idea is the following: choose a subset $\mathbb{M}_r \subset \mathbb{R}^m$ (domain of interest) and, whenever the reported answer $x$ is outside $\mathbb{M}_r$, return the closest point to $x$ in $\mathbb{M}_r$. In the exact semantics the truncation is safe, in the sense that it has been proved that it does not alter differential privacy. However, the fact $n$ and $n'$ are not close means that we cannot expect the properties of $n$ to be transferred to $n'$. Indeed, there are cases in which this method is not safe in the finite-precision semantics, unless we make strong assumptions on the computational errors. Since we are trying to be as general as possible, we cannot use this kind of truncation.[5] Hence, we adopt the following simple approach: Whenever the result is outside the intended subset $\mathbb{M}_r$ of $\mathbb{R}^m$, we return an exception.

This truncation method is quite drastic in the sense that all values outside $\mathbb{M}_r$ are collapsed in the same value (the "exception"). The advantage is that under relatively mild assumptions we can prove that this method is safe in all cases. On the other hand, the loss of utility is higher than in the standard method mentioned above. Better mechanisms can be found, but they are specific to some algorithms or to some dimensions. For instance, we will show later that in the uni-dimensional case ($m = 1$) it is possible to return an extremal value.

We denote by $\infty$ the exception, i.e., the value returned by the mechanism when $f(D) + n(U) \notin \mathbb{M}_r$. Hence, the truncated mechanism $\mathcal{K}_{\mathbb{M}_r}$ is defined as:

**Mechanism 3.**

$$\mathcal{K}_{\mathbb{M}_r}(D) = \begin{cases} \mathcal{K}_n(D) & \textit{if } \mathcal{K}_n(D) \in \mathbb{M}_r \\ \infty & \textit{otherwise} \end{cases}$$

We now present the conditions that ensure the the above mechanism is safe. The first one is that for a suitable $\mathbb{M}_r \subset \mathbb{R}^m$ there exist a domain $U_r \subset [0, 1[^m$ such that

[5]Note that redrawing a new random value, when the result is outside $\mathbb{M}_r$, would not be a solution, because it would change the distribution in a way that may cause the loss of differential privacy.

$n$ and $n'$ are close in $U_r$, while outside $U_r$ the result determined by $n$ falls out of (the expansion of) $\mathbb{M}_r$, and hence gets truncated.

**Condition 1.** *For some $k$ and $\delta_n$ in $\mathbb{R}^+$, and sets $U_r \subset [0,1[^m$ and $\mathbb{M}_r \subset \mathbb{R}^m$*

(i) *$n$ and $n'$ are $(k, \delta_n)$-close on $U_r$*
(ii) $\forall u \notin U_r,\ f(D) + n(u) \notin \mathrm{expand}(\mathbb{M}_r, +k\delta_0 + \delta_n)$

The above condition may look strong, but in fact it is a quite reasonable assumption. For instance, it is granted when $\|f\|_\infty < \infty$ and $n$ is absolutely continuous. In such case, indeed, we can construct $k$, $\delta_n$, $U_r$ and $\mathbb{M}_r$ in the following way.

**Construction 3.1.** *First choose $U_r$ to be a closed subset of $[0,1[^m$.[6] Then set $k$ to be the maximal value for the derivative of $n$ in $U_r$, whose existence is ensured by the absolute continuity of $n$, and set $\delta_n$ to be a bound for $\|n - n'\|_\infty$ in $U_r$.* [7] *Finally, choose $\mathbb{M}_r$ to be the largest set that satisfies $U_r \subseteq n^{-1}(\mathrm{expand}(\mathbb{M}_r, +\|f\|_\infty + k\delta_0 + \delta_n))$.*

**Proposition 3.2.** *$k$, $\delta_n$, $U_r$ and $\mathbb{M}_r$ defined in Construction 3.1 satisfy Condition 1.*

**Proof** The first part of Condition 1 follows from Proposition 3.1. For the second part, let $f(D) + n(u) \in \mathrm{expand}(\mathbb{M}_r, +k\delta_0 + \delta_n)$. Then $n(u) \in \mathrm{expand}(\mathbb{M}_r, +\|f\|_\infty + k\delta_0 + \delta_n)$ and therefore $u \in \mathbb{M}_r$. □

Previous condition ensures that the implementation of the noise behaves well inside $U_r$, and that outside $U_r$ we don't need to worry about the value of $n$ because the result (in the ideal semantics) gets truncated. However, we still need to worry about the result of $n'$ outside $U_r$, because nobody ensures that the result gets truncated also in the implementation. More precisely, it could be that for some value outside $U_r$, $n'$ returns a value inside $\mathbb{M}_r$. To avoid this, we require implementation to be monotonic with respect to the ideal semantics (Condition 2).

Another way to understand the necessity of the monotonicity condition is the following: If we do not require this property, we could have an implementation of $n'$ such that for all $u \notin \mathbb{M}_r$, $n'(u) = 0$. Such an implementation would be valid with respect to Condition 1 because we do not ask for closeness of $n$ and $n'$ outside of $U_r$. Then, for all $u \notin U_r$, $\mathcal{K}'_{\mathbb{M}_r}(D)(u) = f'(D)$ will not be truncated if the domain of $f'$ is a subset of $\mathbb{M}_r$ and therefore the probability that the returned answer by $\mathcal{K}'_{\mathbb{M}_r}$ is just the true answer become much higher than expected.

**Condition 2.** *The implementation respects the order. Namely, for every function $g : \mathbb{R}^j \to \mathbb{R}^k$ and its implementation $g'$, for all $x \in \mathbb{R}^j$ and $y \in \mathbb{R}^j$, $g(x)_i \leq g(y)_i$ implies $g'(x)_i \leq g'(y)_i$ where for $i \in [1,k]$ $z_i$ is the projection of $z$ on the $i^{th}$ component.*

Finally, the algorithm makes a test to check whether the result is in $\mathbb{M}_r$. Let $t$ be the test function, i.e., $t(x) = 1$ if $x \in \mathbb{M}_r$ and $t(x) = 0$ otherwise. The implementation

[6] The choice of $U_r$ influences the utility and the optimal set could be obtained by a fixpoint construction, but this is out of the scope of this paper.

[7] When $n$ is a well-known function, often there exist implementations which are full precision, in the sense that the error is only due to the fact the result has been rounded to the nearest representable number. In such case computing $\delta_n$ is easy.

$t'$ in general does not coincide with $t$, and therefore it accepts a set $\mathbb{M}_r{}' = t'^{-1}(\{1\})$ different from $\mathbb{M}_r$. Hence, we need to ensure that the implementation of the test is safe, namely that in the implementation we do the truncation every time that we would do it in the ideal semantics. This can be ensured with the following condition.

**Condition 3.** *The test $t'$ is such that for all $x \in \mathbb{R}^m$, $t(x) = 0$ implies $t'(x) = 0$ i.e. $\mathbb{M}_r{}' \subseteq \mathbb{M}_r$.*

We are now able to provide a bound on the maximal error of the implementation. Note that the error between $n$ and $n' \circ n_0(u)$ can be bound only inside $U_r$, but it will be convenient to consider instead a variant of the implementation, $\tilde{n}' \circ \tilde{n_0}$, which coincides with $n' \circ n_0(u)$ inside $U_r$ and coincides with $n$ outside. The advantage is that $\tilde{n}' \circ \tilde{n_0}$ is close to $n$ in all the domain. From the point of view of the implementation of the whole mechanism it is not a problem to replace $n' \circ n_0(u)$ by $\tilde{n}' \circ \tilde{n_0}$, because outside $U_r$ the result is truncated anyway.

**Definition 3.3.** *The mechanism $\tilde{\mathcal{K}}'_n$ is defined as $\tilde{\mathcal{K}}'_n(D) = f'(D) + \tilde{n}' \circ \tilde{n_0}(U)$, where $\tilde{n}'$, $\tilde{n}_0$ are defined so to satisfy the following:*

$$\tilde{n}' \circ \tilde{n_0}(u) = \begin{cases} n' \circ n_0(u) & \textit{if } u \in U_r \\ n(u) & \textit{otherwise} \end{cases}$$

**Proposition 3.3.** *When Conditions 1, 2 and 3 hold, the implementation $\mathcal{K}'_{\mathbb{M}_r{}'}$ of Mechanism 3 satisfies the following:*

$$\mathcal{K}'_{\mathbb{M}_r{}'}(D) = \begin{cases} \tilde{\mathcal{K}}'_n(D) & \textit{if } \tilde{\mathcal{K}}'_n(D) \in \mathbb{M}_r{}' \\ \infty & \textit{otherwise} \end{cases}$$

*and $\tilde{\mathcal{K}}'_n$ is $\delta_t$-close to the mechanism in the ideal semantics, in the sense that*

$$\forall u \in [0,1[^m, d(f'(D) + n(u), f'(D) + \tilde{n}' \circ \tilde{n_0}(u)) \leq \delta_t$$

*where $\delta_t = k\delta_0 + \delta_n$.*

**Proof** By Condition 1 (ii) we have $\forall D, \forall u \notin U_r, f(D) + n(u) \notin \mathrm{expand}(\mathbb{M}_r, +k\delta_0 + \delta_n)$. Then by Condition 1 (i) and Condition 3 we have $\mathbb{M}_r{}' \subseteq f(D) + n' \circ n_0(U_r)$. By Condition 2 we derive $\forall u \notin U_r, f(D) + n' \circ n_0(u) \notin \mathbb{M}_r{}'$. Since $n' \circ n_0(u)$ and $\tilde{n}' \circ \tilde{n_0}$ coincide on $U_r$, we have that for all $u \in U_r, t'(f(D) + n' \circ n_0(u)) = t'(f(D) + \tilde{n}' \circ \tilde{n_0})$. Moreover $\tilde{\mathcal{K}}'_n(D)(u) \in \mathbb{M}_r{}'$ implies $u \in U_r$ and therefore $f'(D) + \tilde{n}' \circ \tilde{n_0}(U) = f'(D) + n' \circ n_0(U)$. So the first part of the proposition is proven.

For the second part, we recall that $\tilde{n}'$ and $n$ are $(k, \delta_n)$-close on $U_r$. Therefore by composition of closeness with $n_0$, and since $\tilde{n}' \circ \tilde{n_0} = n' \circ n_0$ on $U_r$, we have the property on $U_r$. Then, since $\tilde{n}' \circ \tilde{n_0} = n$ on $[0,1[^m \backslash U_r$, we have the property on the whole domain $[0,1[^m$.

□

## 4. Quantification of the loss of differential privacy due to computational errors

In the previous section we have provided a bound $\delta_t$ on the computational error. In this section, we use this bound to quantify the loss of differential privacy. First, we explain how to get bounds on probabilities from bounds on computational errors. Then, we show that Mechanism 3 is still not safe and we propose a solution. Finally we show that the implementation of the resulting mechanism is differentially private, although there is a degradation in the level of privacy with respect to the original (ideal) one, and we provide a bound on such degradation.

### *4.1. Modeling the error as a distance between distributions*

Given that we are in a probabilistic setting, it is not so useful to measure the errors due to the finite representation in terms of numerical difference. We should rather measure them in terms of distance between the theoretical distribution and the actual distribution. In this way, we will be able to establish a link with differential privacy, which is indeed, basically, a notion of distance between distributions.

In the literature one can find many definitions of distance between distributions. Here we consider the $\infty$-Wassertein distance [3] since, as we will see, it has a direct relation with differential privacy. We will use $\Gamma(\mu, \nu)$ to denote the set of all measures on $\mathbb{R}^m \times \mathbb{R}^m$ with marginal $\mu$ and $\nu$ respectively. Namely, every $\gamma \in \Gamma(\mu, \nu)$ is a measure such that

$$\gamma(S \times \mathbb{R}^m) = \mu(S) \text{ (left marginal)} \quad \text{and} \quad \gamma(\mathbb{R}^m \times S) = \nu(S) \text{ (right marginal)}$$

We also denote by $\mathrm{supp}(\gamma)$, the set of points where $\gamma$ is non zero.

**Definition 4.1** ($\infty$-Wassertein distance [3])**.** *Let $\mu$, $\nu$ be two probability measures on $(\mathbb{R}^m, \mathcal{S})$ for which there exists a compact $C$ such that $\mu(C) = \nu(C) = 1$ (i.e, the support of $\mu$ and $\nu$ is bounded). The $\infty$-Wassertein distance between $\mu$ and $\nu$ is defined as:*

$$W_\infty(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \left( \sup_{(x,y) \in \mathrm{supp}(\gamma)} d(x, y) \right)$$

We extend this definition to any pair of measures $\mu$ and $\nu$ that differs only on a compact ($\mathbb{M}_r$ in our case) by restricting $\Gamma(\mu, \nu)$ to be null on all set $X \times Y$ such that $X \cap Y = \emptyset$, $X \subset \mathbb{R}^m \setminus \mathbb{M}_r$ and $Y \subset \mathbb{R}^m \setminus \mathbb{M}_r$.

We can now quantify the distance between two random variables:

**Proposition 4.1.** *Let $X$ and $X'$ be two random variables $\Omega \to \mathbb{R}^m$ with distribution $\mu$ and $\nu$ respectively. We have that $\|X - X'\|_\infty \leq \delta$ implies $W_\infty(\mu, \nu) \leq \delta$.*

**Proof** Let $A, B$ be two measurable sets distant by more than $\delta$, namely: $\forall a \in A, b \in B, d(a, b) > \delta$. Then $\|X - X'\|_\infty \leq \delta$ implies that there is no $\omega \in \Omega$ such that $X(\omega) \in A$ and $X'(\omega) \in B$. Hence $P(X \in A \wedge X' \in B)$ is null which means that the support of $(\mu, \nu)$ does not include any such pair of sets $(A, B)$, and therefore we can construct a $\gamma \in \Gamma(\mu, \nu)$ such that $\sup_{(x,y) \in \mathrm{supp}(\gamma)} d(x, y) \leq \delta$. □

The following proposition allows us to use the $\infty$-Wassertein distance between $\mu$ and $\nu$ to establish a bound on $\mu$ in terms of $\nu$.

**Proposition 4.2.** *Let $\mu, \nu$ be two probability measures on $\mathbb{R}^m$. Then:*

$$W_\infty(\mu,\nu) \leq \delta \implies \forall S \in \mathcal{S}, \mu(\mathrm{expand}(S,-\delta)) \leq \nu(S) \leq \mu(\mathrm{expand}(S,+\delta))$$

**Proof** Let $\gamma \in \Gamma(\mu,\nu)$. Since $\nu$ is the right marginal of $\gamma$, $\nu(S) = \int_{\mathbb{R}^m \times S} d\gamma(x,y)$. Since $\gamma(x,y) = 0$ if $d(x,y) > \delta$, we derive $\nu(S) = \int_{\mathrm{expand}(S,+\delta)\times S} d\gamma(x,y)$. Therefore, $\nu(S) \leq \int_{\mathrm{expand}(S,+\delta)\times \mathbb{R}^m} d\gamma(x,y)$. The last expression is the left marginal of $\gamma$ in $\mathrm{expand}(S,+\delta)$, hence by definition: $\nu(S) \leq \mu(\mathrm{expand}(S,+\delta))$.

The other inequality is obtained by replacing $S$ with $\mathbb{R}^m \setminus S$. □

We can now provide upper and lower bounds on the probabilities of the implemented noise in terms of those in the exact semantics and the computational error $\delta_t$:

**Corollary 4.1.** *Let $\mu, \nu$ be the probability measures of $n(U)$, $\tilde{n}' \circ \tilde{n_0}(U)$, respectively. Namely, for all $S \in \mathcal{S}$ (where we recall that $\mathcal{S}$ is the set of measurable sets in $[0,1[^m$), $\mu(S) = P[n(U) \in S]$ and $\nu(S) = P[\tilde{n}' \circ \tilde{n_0}(U) \in S]$. Then, for all $S \in \mathcal{S}$:*

$$\mu(\mathrm{expand}(S,-\delta_t)) \leq \nu(S) \leq \mu(\mathrm{expand}(S,+\delta_t))$$

**Proof** By Proposition 3.3 and Proposition 4.1 it follows that $W_\infty(\mu,\nu) \leq \delta_t$. Then apply Proposition 4.2. □

However, used carelessly, the lower bound provided by the previous corollary is not accurate enough. There is in fact a problem when $S$ is too small: $\mathrm{expand}(S,-\delta_t)$ can be much smaller than $S$ or even be the empty set. Consequently, the probability of the returned answer to be in $\mathrm{expand}(S,-\delta_t)$ will be close to 0, or even 0. Since differential privacy is about ratio, this case would not have a suitable bound. Note that this is exactly the problem described in Example 1.1.

### *4.2. Rounding the answer*

In the last paragraph of previous section we pointed out a problem for differential privacy that may arise when the analyst measure sets that are too small. The most natural way to solve this problem consists in rounding the answer, which has the effect of collapsing all values from a neighborhood into a unique value. In this approach, once the computation of $\mathcal{K}_{\mathbb{M}_r}(D)$ is achieved, we do not return the answer but a "rounding" of the answer through a function $rnd$, whose implementation $rnd'$ is defined as any measurable function that maps all values of $\mathbb{M}_r$ into some finite subset $F$ of $\mathbb{M}_r$, such that $\forall x \in F, rnd'(x) = x$ and such that $rnd'(\infty) = \infty$.

If two values close to the boundary of $\mathbb{M}_r$ are rounded to the same one but one is truncated while not the other one, some problematic behaviors can happen. To avoid that we add an additional condition about $rnd'$ and $\mathbb{M}_r{}'$.

**Condition 4.** *The function $rnd'$ is such that $rnd'^{-1}(F) = \mathbb{M}_r{}'$.*

**Remark 1.** *To achieve this property we assume we already get $\mathbb{M}_{r0}$ that have all other properties (it may have been obtained with the method in Construction 3.1) and that $rnd'$ is a rounding function for $\mathbb{M}_{r0}$ which maps to $F_0$. Then we build $\mathbb{M}_r \subseteq \mathbb{M}_{r0}$ that have Condition 4. We define $L = \max_{x\in F_0} \varnothing(rnd'^{-1}(x))$. Then Condition 4 hold for any set $\mathbb{M}_r{}' = rnd'^{-1}(\mathbb{M}_{r1})$ where $\mathbb{M}_{r1} \subseteq \mathrm{expand}(\mathbb{M}_{r0}{}', -L))$.*

**Proof** We have at first $rnd'$ such that all values of $\mathbb{M}_{r0}$ maps to $F_0$, Since $\mathbb{M}_{r1} \subseteq \text{expand}(\mathbb{M}_{r0}', -L))$ we have $rnd'^{-1}(\mathbb{M}_{r1}) \subseteq \mathbb{M}_{r0}$ $\mathbb{M}_r{}' \subseteq \mathbb{M}_{r0}$ and so $rnd'^{-1}(\mathbb{M}_r{}') \subseteq \mathbb{M}_{r0}$. We consider $F = rnd'(\mathbb{M}_{r1})$, since $\mathbb{M}_{r1} \subseteq \mathbb{M}_{r0}$, $rnd'$ satisfies $\forall x \in F, rnd'(x) = x$ and all values of $\mathbb{M}_r{}'$ maps to $\mathbb{M}_{r1}$ $rnd'$ satisfies all other conditions.

On the other hand, Condition 4 can be rewritten as: $rnd'^{-1}(rnd(\mathbb{M}_{r1})) = rnd'^{-1}(\mathbb{M}_{r1})$. Since $rnd'(\mathbb{M}_{r1}) \subseteq \mathbb{M}_{r1}$ we have $rnd'^{-1}(rnd(\mathbb{M}_{r1})) \subseteq rnd'^{-1}(\mathbb{M}_{r1})$. For the other inclusion, if $x \in rnd'^{-1}(\mathbb{M}_{r1})$, $rnd'(rnd'(x)) \in \mathbb{M}_{r1}$ since $rnd'$ is idempotent on $\mathbb{M}_{r0}$ and $rnd'^{-1}(\mathbb{M}_{r1}) \subseteq \mathbb{M}_{r0}$. This means that $x \in rnd'^{-1}(rnd'(\mathbb{M}_{r1}))$. □

**Mechanism 4.** *Our final mechanism is $\text{rnd}(\mathcal{K}_{\mathbb{M}_r})$, where $\text{rnd}$ is a rounding function.*

Although the sets corresponding to the elements of $F$ form a finite collection, it will be convenient, for the sake of uniformity, to consider them as the generators of a $\sigma$-algebra. Hence, we define $\mathcal{S}_0' = \{rnd'^{-1}(x) | x \in F\}$, and $\mathcal{S}'$ as the $\sigma$-algebra generated by $\mathcal{S}_0'$. Our mechanism allows us to restrict our analysis of probabilities only to $\mathcal{S}'$.

**Proposition 4.3.** $\forall S \in \mathcal{S}', P[\mathcal{K}'_{\mathbb{M}_r{}'}(D_1) \in S] \leq \exp(\epsilon') P[\mathcal{K}'_{\mathbb{M}_r{}'}(D_2) \in S]$ *implies* $rnd'(\mathcal{K}'_{\mathbb{M}_r{}'})$ *is $\epsilon'$-differentially private :*
$\forall S \in \mathcal{S}, P[rnd'(\mathcal{K}'_{\mathbb{M}_r{}'})(D_1) \in S] \leq \exp(\epsilon') P[rnd'(\mathcal{K}'_{\mathbb{M}_r{}'})(D_2) \in S]$.

**Proof** The image of $\mathcal{K}'_{\mathbb{M}_r{}'}$ is $\mathbb{M}_r{}' \cup \infty$ so the image of $rnd'(\mathcal{K}'_{\mathbb{M}_r{}'})$ is $F \cup \{\infty\}$. Therefore, for any set $S \in \mathcal{S}$, $P(rnd'(\mathcal{K}'_{\mathbb{M}_r{}'}) \in S) = P(rnd'(\mathcal{K}'_{\mathbb{M}_r{}'}) \in S \cap (F \cup \{\infty\}))$. Which is equivalent to , $P(rnd'(\mathcal{K}'_{\mathbb{M}_r{}'}) \in S) = P(rnd'(\mathcal{K}'_{\mathbb{M}_r{}'}) \in (S \cap F) \cup (S \cap \{\infty\}))$. This can be rewritten as $P(rnd'(\mathcal{K}'_{\mathbb{M}_r{}'}) \in S) = P(\mathcal{K}'_{\mathbb{M}_r{}'} \in rnd'^{-1}(S \cap F) \cup rnd'^{-1}(S \cap \{\infty\}))$. Since Condition 4 ensures that $rnd'^{-1}(F) \subseteq \mathbb{M}_r{}'$, for all $S \subseteq F$, $rnd'(\mathcal{K}'_{\mathbb{M}_r{}'})^{-1}(rnd^{-1}(S)) = \mathcal{K}'_{\mathbb{M}_r{}'}{}^{-1}(rnd^{-1}(S))$. Then since Condition 4 ensures that $rnd'^{-1}(F) \supseteq \mathbb{M}_r{}'$, $rnd'(\mathcal{K}'_{\mathbb{M}_r{}'})^{-1}(\{\infty\}) = \mathcal{K}'_{\mathbb{M}_r{}'}{}^{-1}(\{\infty\})$. From these two equalities, we get $P(rnd'(\mathcal{K}'_{\mathbb{M}_r{}'}) \in S) = P(\mathcal{K}'_{\mathbb{M}_r{}'} \in rnd'^{-1}(S \cap F) \cup rnd'^{-1}(S \cap \{\infty\}))$. This can be rewritten as: $P(rnd'(\mathcal{K}'_{\mathbb{M}_r{}'}) \in S) = P(\mathcal{K}'_{\mathbb{M}_r{}'} \in rnd'^{-1}(S \cap (F \cup \{\infty\})))$. Since for all $S \in \mathcal{S}$, $rnd'^{-1}(S \cap (F \cup \{\infty\})) \in \mathcal{S}'$, we get the implication. □

We now quantify the efficiency of the rounding function with respect to helping in establishing bounds on ratios between the probability measures of the implementation. To this purpose, we introduce the notion of ratio between the bounds established in Corollary 4.1. This notion will be used in Theorem 4.1.

**Definition 4.2** (Rounding ratio)**.** *We define $R$ to be the maximal ratio between the areas* $\text{expand}(S, +\delta_t)$ *and* $\text{expand}(S, -\delta_t)$ *over all values that can be returned:*

$$R = \max_{S \in \mathcal{S}_0' \setminus \emptyset} \frac{\lambda(\text{expand}(S, \delta_t) \setminus \text{expand}(S, -\delta_t))}{\lambda(\text{expand}(S, -\delta_t))} \tag{4}$$

*Where $\lambda$ is the Lebesgue measure on $\mathbb{R}^m$.*

The $R$ defined by (4) may look quite hard to compute, however, when $\text{rnd}$ is particularly regular, we can bound $R$ from above:

**Proposition 4.4.** *Assume the rounding function* $\mathrm{rnd}_l : \mathbb{R}^m \to \mathbb{R}^m$ *is defined as* $\mathrm{rnd}_l(x_1, \ldots, x_m) = (\lfloor lx_1 \rfloor / l, \ldots, \lfloor lx_m \rfloor / l)$. *Then,*

$$R \leq \left( \frac{l + 2\delta_t}{l - 2\delta_t} \right)^m - 1$$

**Proof** With such a rounding, the space is split in hypercubes $C$ of uniform size $l$. Therefore $\mathrm{expand}(C, \delta_t)$ is included in a hypercube of size $l+2\delta_t$ and $\mathrm{expand}(C, -\delta_t)$ is exactly a hypercube of size $l - 2\delta_t$. The Lebesgue measure of an hypercube of size $s$ is by definition $s^m$. Hence the measure of $\mathrm{expand}(C, -\delta_t)$ is $(l - 2\delta_t)^m$ and the measure of $\mathrm{expand}(C, \delta_t) \setminus \mathrm{expand}(C, -\delta_t)$ is $(l + 2\delta_t)^m - (l - 2\delta_t)^m$. □

*4.3. Strengthening the differential privacy property*

The last issue, in order to be able to quantify the loss due to the finite precision, comes from the definition of differential privacy. This definition relies on the definition of sensitivity in the exact semantics while here we need the sensitivity in the finite precision semantics. More precisely, if the noise has a density function $p$, then differential privacy can be expressed by the following inequality.

$$\forall x, y \in \mathbb{R}^m, d(x, y) \leq \Delta_f \implies p(x) \leq \exp(\epsilon) p(y) \tag{5}$$

In the past sections we have been able to establish a relation between the implemented noise $p'$ and the ideal noise $p$, but we do not know anything about the relation between the implemented sensitivity $\Delta_{f'}$ and the ideal one, $\Delta_f$, so we cannot use (5) directly to derive conclusions about the level of privacy satisfied by the implementation.

There are several ways to solve this problem. Here, we choose to constrain the distribution of $X$. Indeed, when using a Laplace distribution (which is the most used, adding this constraint allows us to keep the bound as accurate as if there was no problem about the distribution.

**Condition 5.** *Given a mechanism* $\mathcal{K}(D) = f(D) + X$ *we say that* $\mathcal{K}$ *satisfies condition 5 with parameter* $\epsilon$ *(the desired parameter of differential privacy) if the random variable* $X$ *has a probability distribution which is absolutely continuous according to the Lebesgue measure, and*

$$\forall S \in \mathcal{S}, r_1, r_2 \in \mathbb{R}^m, P[r_1 + X \in S] \leq \exp(\epsilon d(r_1, r_2) / \Delta_f) P[r_2 + X \in S]$$

This property is actually stronger than differential privacy as we state in the following proposition.

**Proposition 4.5.** *Condition 5 implies that the mechanism* $\mathcal{K}(D) = f(D) + X$ *is* $\epsilon$*-differentially private.*

**Proof** Let $D_1$ and $D_2$ be two databases such that $D_1 \sim D_2$. Let $r_1 = f(D_1)$ and $r_2 = f(D_2)$ be two answers. By definition of sensitivity, $d(r_1, r_2) \leq \Delta_f$ so $\exp(\epsilon^{d(r_1, r_2)} / \Delta_f) \leq \exp(\epsilon)$. Hence, $P[\mathcal{K}(D_1) \in S] \leq \exp(\epsilon) P[\mathcal{K}(D_2) \in S]$. □

We chose this condition because in case the noise has a Laplacian distribution then the converse of Proposition 4.5 holds and therefore in this case Condition 5 and differential privacy are equivalent.

**Proposition 4.6.** *Let $\mathcal{K}(D) = f(D) + X$ be a mechanism, and assume that $X$ is Laplacian. If $\mathcal{K}$ is $\epsilon$-differentially private (w.r.t. $f$), then Condition 5 holds.*

**Proof** First, we show that if $\mathcal{K}$ is $\epsilon$-differentially private then $b \leq \epsilon/\Delta_f$ holds for the scale parameter $b$ of $X$. Let $D_1 \sim D_2$ with $d(f(D_1), f(D_2)) = \Delta_f$. By $\epsilon$-differential privacy we have, for any $S \in \mathcal{S}$: $P[f(D_1) + X \in S] \leq \exp(\epsilon) P[f(D_2) + X \in S]$. From the density function of the Laplace noise (Definition 2.6), we derive: $K(n, d)d\lambda \leq \exp(\epsilon) K(n, d) \exp(-b\Delta_f) d\lambda$. Hence,

$$b \leq \epsilon/\Delta_f. \tag{6}$$

Now, by definition of the density function, we have:
$P[r_2 + X \in S] = \int_{x \in S} K(n, d) \exp(-bd(x, r_2)) d\lambda$. From the triangular inequality, we derive: $P[r_2 + X \in S] \geq \int_{x \in S} K(n, d) \exp(-b(d(x, r_1) + d(r_1, r_2))) d\lambda$. Hence, $P[r_2 + X \in S] \geq \exp(-bd(r_2, r_1)) \int_{x \in S} \exp(-bd(r_1, x)) d\lambda$. From inequality (6), we derive: $P[r_2 + X \in S] \geq \exp(-\epsilon d(r_2, r_1)/\Delta_f) \int_{x \in S} \exp(-bd(r_1, x)) d\lambda$. Finally, $P[r_2 + X \in S] \geq \exp(-\epsilon d(r_2, r_1)/\Delta_f) P[r_1 + X \in S]$. □

*4.4. Preserving differential privacy*

Now we have established all necessary conditions and we can finally state our main theorem.

**Theorem 4.1.** *If mechanism $rnd(\mathcal{K}_{\mathbb{M}_r})$ (Mechanism 4) has an implementation $rnd'(\mathcal{K}'_{\mathbb{M}_{r'}})$ such that conditions 2, 1, 3 and 4 hold and the density function of $n(U)$ satisfies condition 5, then the implemented mechanism is $\epsilon'$-differentially private, namely:*

$$\forall S \in \mathcal{S}, P[rnd(\mathcal{K}_{\mathbb{M}_r})'(D_1) \in S] \leq \exp(\epsilon') P[rnd(\mathcal{K}_{\mathbb{M}_r})'(D_2) \in S]$$

*where*

$$\epsilon' = \epsilon + \ln(1 + R \exp(\epsilon(L + \delta_t)/\Delta_{f'}))$$

*with $R$ corresponds to Definition 4.2, $L = \max_{S \in \mathcal{S}'_0} \varnothing S$ and $\delta_t$ as defined in Proposition 3.3.*

**Proof** According to Proposition 4.3, we only have to consider the Mechanism 2 on the sigma algebra $\mathcal{S}'$. Moreover since $\mathcal{S}'$ is generated by a finite number of disjoint sets, we only have to prove the property for all $S \in \mathcal{S}'_0$. In fact we will prove the property only for all $S \in \mathcal{S}'_0 \setminus \{\infty\}$. Indeed, once this has been proved the fact that $P(rnd(\mathcal{K}_{\mathbb{M}_r})'(D_i) \in \{\infty\}) = 1 - P(rnd(\mathcal{K}_{\mathbb{M}_r})'(D_i) \in \bigcup_{S \in \{\mathcal{S}'_0 \setminus \{\infty\}\}} S)$ implies the property for $\{\infty\}$.

So, we consider $S \in \mathcal{S}'_0 \setminus \{\infty\}$. In that case, $\mathcal{K}'_{\mathbb{M}_{r'}} = \tilde{\mathcal{K}}'_n$. We define $P_1 = P[\tilde{\mathcal{K}}'_n(D_1) \in S]$ and $P_2 = P[\tilde{\mathcal{K}}'_n(D_2) \in S]$. Since $\nu$ is the measure associated to $\tilde{n}' \circ \tilde{n_0}(U)$, we have $P_i = \nu(S' - f'(D_i))$.

With Proposition 3.3 and Theorem 4.1, we get $d(\nu, \mu) \leq \delta_t$.

From Proposition 4.2 we derive $P_1 \leq \mu(\text{expand}(S, \delta_t) - f'(D_1))$ and $P_2 \geq \mu(S_2)$ where $S_2 = \text{expand}(S, -\delta_t) - f'(D_2)$. In the following, we denote by $S_1$ the

set $\text{expand}(S, \delta_t) \setminus \text{expand}(S, -\delta_t)) - f'(D_1$. The additivity property of measures grants us $\mu(\text{expand}(S, \delta_t) - f'(D_1)) = \mu(\text{expand}(S, -\delta_t) - f'(D_1)) + \mu(S_1)$. Condition 5 can be expressed in term of the measure as: $\forall S \in \mathcal{S}, r \in \mathbb{R}^m, \mu(S) \leq \exp\left(\epsilon \|r\| / \Delta_{f'}\right) \mu(S - r)$. From this inequality and the one of $P_1$, we can derive, since $\|f'(D_2) - f'(D_1)\| \leq \Delta_{f'}$: $P_1 \leq \exp(\epsilon) P_2 + \mu(S_1)$.

Since the probability is absolutely continuous according to the Lebesgue measure (Condition 5), we can express the probability with a density function $p$: $\forall S \in \mathcal{S}$, $\mu(S) = \int_S p(x) d\lambda$. We derive: $\forall S \in \mathcal{S}, \min_{x \in S} p(x) \leq \mu(S)/\lambda(S)$. By applying this property on $S_2$, we get: $\min_{x \in S_2} p(x) \leq \mu(S_2)/\lambda(S_2)$. Since the set is compact, the minimum is reached : $\exists x_0 \in S_2, p(x_0) \leq P_2/\lambda(S_2)$. By a triangular inequality, we get : $\forall x \in S_1, d(x, x_0) \leq \Delta_{f'} + L + \delta_t$. Hence, from Condition 5 we derive: $\forall x \in S_1, p(x) \leq \exp\left(\epsilon(\Delta_{f'} + L + \delta_t)/\Delta_{f'}\right) p(x_0)$. Then since $\mu(S_1) = \int_{S_1} p(x) d\lambda$: $\mu(S_1) \leq \exp(\epsilon(\Delta_{f'} + L + \delta_t)/\Delta_{f'}) \lambda(S_1)/\lambda(S_2) P_2$. Because the Lebesgue measure is invariant by translation, we can rewrite this inequality with $R$ from Definition 4.2: $\mu(S_1) \leq \exp(\epsilon(\Delta_{f'} + L + \delta_t)/\Delta_{f'}) R P_2$. Finally, since we already get $P_1 \leq \exp(\epsilon) P_2 + \mu(S_1)$, we obtain : $P_1 \leq (1 + R \exp(\epsilon L + \delta_t / \Delta_{f'})) \exp(\epsilon) P_2$.

□

## 5. Application to the Laplacian noise in one dimension

In our main theorem 4.1, we state a general result which is parametric in the dimension $m$ of the range space, in the law $n(U)$ for the added noise, and on the implementation of this noise. Now, we consider the original case for which differential privacy has been used, i.e. the case where $m = 1$ and $n(U)$ is distributed according to the standard linear Laplacian. This specific case will allow us to quantify the loss numerically. We show that the proposed mechanism 4 is still unsafe. Indeed, as we show, a set $\mathbb{M}_r$ that would prevent a large error $\delta_t$ would be too small to be useful. To solve this problem, we propose an improvement on the implementation of the random noise and show that, in this new setting, the loss is negligible.

### *5.1. Requirements and architecture assumptions*

To be able to give numerical result we need to define which mechanism we use and where it is implemented.

*Differential privacy mechanism.* We consider some intended degree of privacy $\epsilon$. We consider the function $f$ which ranges over some interval $\mathbb{M}_r = [m, M]$. We denote by $r = M - m$ the size of this interval. We also express the sensitivity as a function of a parameter $N$: $\Delta_f = r/N$. If the query is the summation of some subset of the entries, $N$ represents the minimal number of entries that will be in the subset by the query.

A Laplace noise that provides $\epsilon$-differential privacy in the exact semantics must have scale parameter $\Delta_f/\epsilon$. Now, to respect the pattern of mechanism 4, we need to decide the truncation range and the rounding function. Since $f$ ranges over $[m, M]$, we truncate the result and return an error if we output a result outside of this range. For this example, we decide to round the result so that the result is a multiple of $r/2^{52-s}$ where $s$ is an integer representing the number of significant digits that are removed.

*Assumptions about the machine architecture.* We consider that the mechanism is implemented into a machine with the floating-point architecture following the IEEE standard [12] on 64 bits.

We assume that the uniform generator used here returns a multiple of $2^{-53}$.

The next step consists of picking a number according to the Laplace distribution. The easier way to achieve this is to use the following formula.

$$X = n(U) = -b\,\mathrm{sgn}(U - 1/2)\,\ln(1 - 2|U - 1/2|)$$

Indeed, if $U$ is a random uniform variable then $n(U)$ is a centered Laplacian distribution with scale parameter $b$. This standard technique to get a Laplace distribution is convenient since its implementation just uses arithmetic operators, absolute values and the logarithm function that can be implemented without any loss of precision [10].

$$n(u) = {}^{\Delta_f}/_{\epsilon}\mathrm{sgn}(u - 1/2)\ln(1 - 2|u - 1/2|). \tag{7}$$

The computation of the logarithm is in two steps. First, from the representation of $u = u_m 2^{u_e}$ where $u_m$ is the mantissa and $u_e$ the exponent of $u$, we have $\ln(u) = u_e \ln(2) + \ln(u_m)$. Hence, we reduce the problem to computing the logarithm for numbers in $[1, 2[$. This algorithm has been proved full-precision for some implementations [13], meaning that the error made by the algorithm is just the one due to the finite representation.

### *5.2. Weakness of the mechanism*

*Initial uniform noise.* As we explained in the subsection 3.3, since the uniform random generator returns numbers that are multiple of $2^{-53}$, the $\delta_0$ parameter defined in definition 3.1 is $\delta_0 = 2^{-53}$.

*Closeness of $n$ and $n'$.* In order to apply our theorem, we need to prove that condition 1 is satisfied. By proposition 3.1, it is sufficient to prove that, in the interval of interest, $n(u)$ is $k$-Lipschitz and that $|n(u) - n'(u)| \leq \delta_n$.

**Proposition 5.1.** *In our case, $n$ is $k$-Lipschitz with*

$$k = \frac{2\Delta_f}{\epsilon}\exp\left(\frac{\epsilon r}{\Delta_f}\right)$$

**Proof** Since the result $n(u)$ is always truncated if the final result of our mechanism is outside of $[m, M]$, this means, since $f(D)$ belongs to $[m, M]$, that the result of $n(u)$ is always truncated when $|n(u)| \geq r = (M - m)$. So we are interested in knowing the $k$ factor for which $n$ is $k$-Lipschitz in $n^{-1}([-2r, 2r])$. Since $n \in C^1$ (its derivative is continuous), it is enough to compute the maximal value of its derivative in this interval. So first we have to compute $n^{-1}([-2r, 2r])$. The equation $|n(u)| = r$ is equivalent to $|{}^{\Delta_f}/_{\epsilon}\mathrm{sgn}\,(u - {}^1/_2)\ln\,(1 - 2\,|u - {}^1/_2|)| = r$. We derive

$$u = {}^1/_2 \pm \left({}^1/_2 - {}^1/_2\exp(-{}^{\epsilon r}/_{\Delta_f})\right). \tag{8}$$

We have $\underset{\cdot}{\mathrm{n}}/\underset{\cdot}{\mathrm{u}} = {}^{\Delta_f}/_{2\epsilon(1-2|u-1/2|)}$. Finally, the derivative is maximal on the limit of the interval computed in (8)

$$\underset{\cdot}{\mathrm{n}}/\underset{\cdot}{\mathrm{u}} \leq {}^{2\Delta_f \exp({}^{\epsilon r}/_{\Delta_f})}/_{\epsilon} = k.$$

So our function $n$ is $k$-Lipschitz for the previously defined value of $k$. □

This factor is too big as we now illustrate in a numerical application. Indeed, as we can see, this factor depends exponentially on ${}^{\epsilon r}/_{\Delta_f}$. Now, in case of a sum query where values are in the interval $[0, \Delta_f]$, if the database contains $N$ rows, the result will be in $[0, N\Delta_f]$. Since we would like the truncation not to truncate any possible true results, we would like $r = N\Delta_f$. Finally, we get $k = 2^{\Delta_f}/_{\epsilon} \exp(\epsilon N)$.

We have now to compute $\delta_t = k\delta_0 + \delta_n$. We have, in our architecture $\delta_0 = 2^{-53}$. Since summation and multiplication of values only generate errors up to their precision and that the log function is full precision, the computational error $\delta_n$ will be less than $2^{-53}{}^{\Delta_f}/_{\epsilon} r$. Therefore it is negligible. Finally, we have $\delta_t \approx 2^{\Delta_f \delta_0}/_{\epsilon} \exp(\epsilon N)$.

*Rounding the result.* The parameter $R$ in Definition 4.2 is

$$R = {}^{4\delta_t}/_{L-2\delta_t}.$$

This implies that, at least, $2\delta_t < L$. On the other hand, $L$ should be not bigger than ${}^{\Delta_f}/_{\epsilon}$ otherwise the impact of the rounding on the utility of the answer would be bigger than the one due to the noise. This means that we have $2\delta_t < {}^{\Delta_f}/_{\epsilon}$. We can expand this inequality to get

$$4^{\Delta_f \delta_0}/_{\epsilon} \exp(\epsilon N) < {}^{\Delta_f}/_{\epsilon},$$

from which we derive $\epsilon N < 51 \ln(2)$. The standard deviation of the exact mechanism is $d = {}^{\Delta_f}/_{\epsilon}$ and corresponds to the average shift between the true value and the returned value, hence, the ratio $D = {}^{d}/_{\Delta_f}$ corresponds to some "relative expected deviation". From the last inequality and the fact that $51 \ln(2) < 36$, we get $D > {}^{N}/_{36}$. Since to be useful the relative deviation should be less than 1, the protocol does not work for query summing more than 36 rows, which is not acceptable ($N$ is normally greater than one hundred).

### *5.3. Improvement of the implementation*

The last mechanism was not able to provide accurate results in a large range. To solve this problem, we need the uniform generator to be able to return numbers smaller than $2^{-53}$. One way to do that consists in generating the mantissa of the number in the classical way and then generating the exponent according to an exponential law as in [15]. One slightly different way to proceed consists in using a formula equivalent to (7): $n(u, v) = {}^{\Delta_f}/_{\epsilon} v \ln(|u|)$, where $v \in \{-1, 1\}$ and $P(v = 1) = 0.5$.

If we decompose $u$ between its mantissa $1 + u_m$ and its exponent $-(1 + u_e)$, which is a positive integer, we obtain $n(u, v) = {}^{v\Delta_f}/_{\epsilon} \ln(|(1 + u_m)2^{-(1+u_e)}|)$. Therefore, we derive $n(u_m, u_e, v) = {}^{v\Delta_f}/_{\epsilon} (\ln(|1 + u_m|) - \ln(2)(1 + u_e))$.

If we want $U$ to be uniform, then the probability to pick $u = (1 + u_m)2^{-(1+u_e)}$ has to be proportional to the interval in the exact semantics which is rounded in $u$ for that $u_m$ has to be uniform and $u_e$ has to be picked according to an exponential law, i.e.

$P(u_e = n+1) = 2^{-(1+u_e)}$. To generate such a distribution, we can, for instance, flip a coin and count the number of heads before getting odd.

Since there is no rounding error when computing integer random values, we only have to consider errors depending on $u_m$. But, now, for $u_m \in [0,1]$, $n(u_m)$ is $\Delta_f/\epsilon$-Lipschitz. Finally, the total error $\delta_t$ will be of the same order as the precision of the returned result, i.e $\delta_t = 2^{-52}r$.

The parameter $R$ defined in 4.2 is now very close to 0: $R = {}^{4\delta_t}/_{L-2\delta_t}$. Since $L = 2^{-52+s}r$ and $\delta_t = 2^{-52}r$, we derive $R = {}^{4}/_{2^s-2}$. Finally the $\epsilon'$ parameter $\epsilon' = \epsilon + \ln(1 + R\exp(\epsilon^{L+\delta_t}/\Delta_f))$ can be rewritten with the computed values

$$\epsilon' = \epsilon + \ln(1 + {}^{4}/_{2^s-2}\exp(\epsilon^{L+\delta_t}/\Delta_f)).$$

If we set $s = 22$, then we have $2^s \gg 1$. By approximating $1 + 2^s \approx 2^s$ we get $\epsilon' \approx \epsilon + \ln(1 + 2^{-s+2}\exp(\epsilon^{2^s r}/2^{52}\Delta_f))$. When we use the parameter $N = {}^{r}/_{\Delta_f}$, $\epsilon' \approx \epsilon + \ln(1 + 2^{-s+2}\exp(\epsilon^{2^s N}/2^{52}))$. Since $N$ is an indicator of the number of the entries in the database, we can assume that $\epsilon N \ll 2^{30}$. So the exponential is approximately equals to 1. Then since $\ln(1+x) \leq x$, and $2^{10} \approx 10^3$ we conclude that $\epsilon' \approx \epsilon + 10^{-6}$.

## 6. Application to the Laplacian noise in two dimensions

One of the nice features of our theorem 4.1 is its ability to deal with multi-dimensional variables. Here, we present the case in which queries return points in a Euclidean plane. This domain is used, for instance, in geo-location (see [1]). We consider the same architecture as in the previous section. The new difficulty, here, is to generate a random variable with a bivariate Laplace distribution. Since our definition of the multivariate Laplacian is not a standard one, there is no generation protocol in the literature. To generate this noise, we can generate it in polar coordinates and then converts it into Cartesian ones.

The probability density function in Cartesian coordinates is

$$p(x,y) = K\exp(b\sqrt{|x-x_0|^2 + |y-y_0|^2})$$

Following [1], we consider a transformation to polar coordinates.

**Proposition 6.1.** *The distribution above can be generated with the following equation:*

$$n(u_r, u_\theta) = (-({}^{W_{-1}(u_r - 1/e)+1}/_{e})\cos(2\pi u_\theta), -({}^{W_{-1}(u_r - 1/e)+1}/_{e})\sin(\theta))$$

*where $W_{-1}$ is the negative Lambert function characterized by $W(x)\exp(W(x)) = x$.*

**Proof** The radial probability density function is expressed by the following formula: $p(r,\theta) = {}^{b^2}/_{2\pi r}\exp(br)$. Hence the cumulative function for the radius is $C_b(r) = 1 - (1+br)\exp(-br)$. To generate the random variable, we have to compute $r = C_b^{-1}(u)$. We have $C_b(r) = u$ is equivalent to $-(1+br)\exp(-(1+br)) = {}^{u-1}/_{b}$. To solve this equation, we need to introduce the Lambert function defined as the reciprocal of the function $f(x) = x\exp(x)$ Since $f$ is not injective, there actually exist two functions

$W_0$ and $W_{-1}$ such that $W(x)\exp(W(x)) = x$. Here we use the $W_{-1}$ negative function defined on $[-1/e, 0[$. Finally, we get the equation $r = -{}^{W_{-1}(u-1/e)+1}/b$. Once we have got the radius, the angle is obtained by multiplying a uniform random variable in $[0, 1[$ by $2\pi$. Finally, we convert $(r, \theta)$ into Cartesian coordinates. $\square$

While Lambert functions are not algebraic, there exist iterative algorithms to compute them with arbitrary precision [5]. The problem we had with the logarithm in the last section is still relevant here. Indeed, even if we can bound the computational errors, the maximal value that the initial generator can provide is not close enough to 1 to return very large values. In order to avoid a too narrow truncation domain, we might add an additional protocol. Here, however, we will neither describe such a protocol, nor actually compute the maximal error that can be expected with such an algorithm. In the following, we just assume that the error is at most some $\delta_r$ value.

*Truncation.* Since most of the time the domain studied is bound (for instance the public transportation of a city is inside the limit of the city), we can do a truncation. However, we recall that our truncation is made for robustness purpose and not just for utility reasons. Hence, if our domain of interest is a circle, we will not choose $\mathbb{M}_r$ to be the same circle because the probability that the truncation would return an exception would be too high (more than one half if the true result is on the circumference).

*Robustness of $n$.* As in the previous section, we do not analyze an actual implementation but we care about the $k$ factor used for Condition 1. First, we analyze for which $k_C(\epsilon, \varnothing(\mathbb{M}_r))$ the function $C_\epsilon^{-1}$ is $k$-Lipschitz in $[0, \varnothing(\mathbb{M}_r)]$. Since $C$ is differential, this question is equivalent to find the inverse of the minimal value taken by its derivative function on the interval $C_\epsilon^{-1}([0, \varnothing(\mathbb{M}_r)])$. By computing this minimum value, we get

$$k_C(\epsilon, \varnothing(\mathbb{M}_r)) = {}^{\exp(\epsilon\varnothing(\mathbb{M}_r))}/_{2\epsilon + r\epsilon^2}.$$

On the other hand, the computation of $\theta$ is just a multiplication by $2\pi$ of the uniform generator hence $k_\theta = 2\pi$. Then, with the conversion $(r, \theta) \mapsto (r\cos(\theta), r\sin(\theta))$ from polar coordinates to Cartesian coordinates we obtain the global $k$ factor: $k = \sqrt{k_C(\epsilon, \varnothing(\mathbb{M}_r))^2 + 2\pi\varnothing(\mathbb{M}_r)}$. Let $\delta_n$ be the distance between $n$ and $n'$, and $\delta_0$ be the error of the uniform generator. From Proposition 3.3 we get:

$$\delta_t = \sqrt{k_C(\epsilon, \varnothing(\mathbb{M}_r))^2 + 2\pi\varnothing(\mathbb{M}_r)}\delta_0 + \delta_n.$$

*Rounding the answer.* We now compute the parameter $R$ in (4.2). The rounding is made in the Cartesian coordinates, hence, the inverse image of any returned value is a square $S$ of length $L$. Note that $\mathrm{expand}(S, \delta_t)$ is included in the square of length $L + 2\delta_t$ and $\mathrm{expand}(S, -\delta_t)$ is a square of length $L - 2\delta_t$. Hence, the ratio value is smaller than $R = ({}^{L+2\delta_t}/_{L-2\delta_t})^2$.

*Differential privacy.* By Theorem 4.1 we get that (the implementation of) our mechanism is $\epsilon'$-differentially private with $\epsilon' = \epsilon + \ln(1 + ({}^{L+2\delta_t}/_{L-2\delta_t})^2 \exp(\epsilon^{L+\delta_t}/_{\Delta_{f'}}))$. Here, as for the previous example, we need to avoid to truncate outside of a very small circle. While we will not provide details here, one way to solve this problem in floating point precision consists in generating all possible floating points numbers instead

of just the ones which are multiples of $2^{-53}$. That way, with an implementation of $W_{-1}$ which is full precision, we can get a much smaller $\delta_0$.

## 7. Conclusion

This paper concerns the influence of rounding errors in mechanisms achieving differential privacy: a security protocol that relies on real-valued random numbers generation. Existing studies on differential privacy focus either only on the mathematical properties or on technical problems which are particular to a given implementation. Here, we have presented a method to quantify the loss of privacy induced by finite precision. It should be possible to apply this method to most kinds of implementation and computer architecture that achieve differential privacy.

With this model, we have been able to prove that the additive Mechanism 1 which is safe from a theoretical point of view breaks the differential privacy once implemented in a finite precision architecture. This loss has two origins: first, extreme values can cuase significant errors, and secondly, locally, last digits can provide fingerprints of the secret. Our solution solves these two problems: the last digits leakage has been fixed by a rounding procedure and the extreme perturbations by raising an error when the result is outside some range of values.

While proposing these fixes, we have done a quantitative analysis to measure the loss of privacy induced by finite-precision representation. We have proved that when the fixes on the mechanism are implemented, the differential privacy parameter is just increased by some additive factor.

To analyze the relevance of our general result, we have applied it for the standard method (Laplacian in one dimension). This strict application was not really satisfactory: in general, the loss is too important. Therefore, we have proposed a last fix where the uniform random generator of Mechanism 2 has been replaced by a dynamic procedure that provides as many random bits as necessary to grant differential privacy.

Finally, we have explained how to implement a bivariate Laplace noise and how to compute its loss once implemented. Here there is no obvious way to allow a random value to be generated in a large range.

## 8. Acknowledgements

This work was partially supported by the MSR-INRIA joint lab, by the European Union 7th FP project MEALS, by the project ANR-12-IS02-001 PACE, and by the INRIA Large Scale Initiative CAPPRIS.

## References

[1] M. Andrés, N. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2013.

[2] G. Barthe, B. Köpf, F. Olmedo, and S. Z. Béguelin. Probabilistic relational reasoning for differential privacy. In *Proceedings of the 39th Annual ACM Symposium on Principles of Programming Languages (POPL)*. ACM, 2012.

[3] T. Champion, L. De Pascale, and P. Juutinen. The $\infty$-wasserstein distance: Local solutions and existence of optimal transport maps. *SIAM Journal on Mathematical Analysis*, 40(1):1–20, 2008.

[4] S. Chaudhuri, S. Gulwani, R. Lublinerman, and S. NavidPour. Proving programs robust. In T. Gyimóthy and A. Zeller, editors, *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13rd European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9, 2011*, pages 102–112. ACM, 2011.

[5] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the lambert w function. In *Advances in Computational Mathematics*, pages 329–359, 1996.

[6] C. Dwork. Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *33rd International Colloquium on Automata, Languages and Programming (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[7] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In S. Halevi and T. Rabin, editors, *In Proceedings of the Third Theory of Cryptography Conference (TCC)*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[8] M. Gaboardi, A. Haeberlen, J. Hsu, A. Narayan, and B. C. Pierce. Linear dependent types for differential privacy. In R. Giacobazzi and R. Cousot, editors, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'13)*, pages 357–370. ACM, 2013.

[9] I. Gazeau, D. Miller, and C. Palamidessi. A non-local method for robustness analysis of floating point programs. In H. Wiklicky and M. Massink, editors, *Proceedings of the 10th Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL 2012)*, volume 85 of *EPTCS*, pages 63–76, 2012.

[10] G. J. Hekstra and E. F. Deprettere. Floating point cordic, 1993.

[11] S.-S. Ho and S. Ruan. Differential privacy for location pattern mining. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS (SPRINGL)*, pages 17–24. ACM, 2011.

[12] IEEE Task P754. *IEEE 754-2008, Standard for Floating-Point Arithmetic*. IEEE, pub-IEEE-STD:adr, Aug. 2008.

[13] K. Kota and J. R. Cavallaro. Numerical accuracy and hardware tradeoffs for cordic arithmetic for special-purpose processors. *IEEE Trans. Comput.*, 42(7):769–779, July 1993.

[14] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In G. Alonso, J. A. Blakeley, and A. L. P. Chen, editors, *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México*, pages 277–286. IEEE, 2008.

[15] I. Mironov. On significance of the least significant bits for differential privacy. In T. Yu, G. Danezis, and V. D. Gligor, editors, *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 650–661. ACM, 2012.

[16] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Proceedings of the 30th IEEE Symposium on Security and Privacy*, pages 173–187. IEEE Computer Society, 2009.

[17] W. Rudin. *Real and Complex Analysis*. McGraw-Hill, 3rd edition, 1986.