On Significance of the Least Significant Bits For Differential Privacy

Ilya Mironov

Abstract

We describe a new type of vulnerability present in many implementations of differentially private mechanisms. In particular, all four publicly available general purpose systems for differentially private computations are susceptible to our attack.

The vulnerability is based on irregularities of floating-point implementations of the privacy-preserving Laplacian mechanism. Unlike its mathematical abstraction, the textbook sampling procedure results in a porous distribution over double-precision numbers that allows one to breach differential privacy with just a few queries into the mechanism.

We propose a mitigating strategy and prove that it satisfies differential privacy under some mild assumptions on available implementation of floating-point arithmetic.

1 Introduction

The line of work on privacy of statistical databases starting with seminal papers by Dwork et al. [DN04, Dwo06, DMNS06] advanced a general approach towards achieving privacy via randomization of outputs. Instead of releasing *accurate* answers to statistical queries, which potentially leads to a breach of privacy, the curator of a database randomizes its responses. For numerical outputs, the randomization process typically involves adding to the accurate answer a secret random number (noise) sampled from a publicly-known distribution, whose shape and parameters are chosen to guarantee a certain level of privacy.

Realizations of differentially private mechanisms include privacy-preserving histograms [MKA$^+$08], logistic regressions [CM08], recommender systems [MM09], network-analysis tools [MM10], and many others. In response to the need for solutions that would simplify development of privacy-preserving systems, with specific focus on cloud computing, several platforms were proposed in recent years: PINQ [McS09], Airavat [RSK$^+$10], Fuzz [HPN11], and GUPT [MTS$^+$12]. These systems differ in their implementation languages, target operating systems, and programming models, and yet they share essentially the same noise-generation mechanism central to differential privacy.

We describe a new type of vulnerability present in all general purpose systems and many implementations of differentially private algorithms available to us. The vulnerability, which can be remarkably effective against differentially private systems, exploits the porous distributions of textbook implementations of the Laplacian mechanism resulting from finite precision and rounding effects of floating-point operations. While differential privacy requires all outputs to be feasible for all possible inputs and having similar probabilities when the inputs are close, the results of floating-point arithmetic will be concentrated on a small subset of outputs. Careful examination of these subsets demonstrate that they overlap only partially on close inputs, thus breaking the guarantee of differential privacy, which would have applied if all operations were computed with infinite precision and unlimited source of entropy.

Our attack defeats systems that were specifically designed to eliminate side-channel vulnerabilities, employing strategies such as LSM-based mandatory access control mechanisms [RSK$^+$10, MTS$^+$12], or a generalized linear-type system [HPN11]. We remark that while these systems carefully block all traditional side channels—timing and state—the output of the differentially private mechanism, which is supposed to be safe to release due to its mathematical properties, creates a side channel of its own. The reason for this exploitable vulnerability is that floating-point implementations of differentially private algorithms deviate in important respects from their mathematical abstractions.

Although workarounds are relatively simple and low overhead, the fact that these independent and very diverse implementations are all susceptible to the same attack, demonstrates that subtle properties of floating-point arithmetic can be easily overlooked.

The paper's organization is as follows. We recall the definition of differential privacy and describe the Laplacian mechanism in Section 2. Platforms for differentially private computations are reviewed in Section 3, followed by discussion of relevant facts of floating-point arithmetic and its implications to implementations of the Laplacian mechanism in Section 4. We describe our main attack in Section 4.5, analyze its success probability in Section 4.6, and discuss its implications for privacy. Section 5 on defense mechanisms considers two flawed approaches, followed by proof of security of a floating-point implementation of the snapping mechanism. Another example of vulnerability due to use of floating-point arithmetic is given in Section 6.

2 Differential Privacy

Differential privacy, introduced by Dwork et al. [Dwo06, DMNS06] and recently surveyed in [Dwo11], is a standard notion of privacy for computations over databases. It is defined for randomized functions (i.e., distributions over the set of possible outputs) and can be formally stated as follows:

> Randomized function $f\colon \mathcal{D} \mapsto \mathcal{R}$ satisfies ϵ-*differential privacy* if for all subsets $S \subset \mathcal{R}$ and for all pairs of *adjacent* inputs $D, D' \in \mathcal{D}$:
>
> $$\Pr[f(D) \in S] \leq e^{\epsilon} \Pr[f(D') \in S].$$

The exact definition of two databases being adjacent is domain-specific and dependent on the desired level of security guarantee. Commonly used definitions are the user-level privacy, where two databases are considered adjacent if they differ in contribution of a single person, and the entry-level privacy, where the protected entity is a single item, action, or attribute.

An equivalent Bayesian interpretation of the definition of differential privacy asserts that if f is ϵ-differentially-private and O is the observed output of f, then for all priors p and all pairs of adjacent D and D'

$$\frac{p(D \mid O)}{p(D' \mid O)} \leq e^{\epsilon} \frac{p(D)}{p(D')}.$$

If two database are adjacent when they differ in records pertaining to one individual, we say that the adversary's probability of guessing whether any one person, chosen *after* interacting with the mechanism, is present in the database increases multiplicatively by at most e^{ϵ} ($\approx 1 + \epsilon$ for small ϵ). This guarantee holds for any auxiliary information, modeled as a prior on the database or that person.

One important consequence of this interpretation is that all outputs of f that have non-zero support in $f(D)$ must also be feasible under $f(D')$. Otherwise, the adversary could rule out D' for some outputs in violation of the definition of differential privacy.

If the function of interest is not differentially private, such as when it is deterministic, it can be approximated by a differentially private function. Minimizing the loss of accuracy due to approximation, or the noise level, is the major research goal in differential privacy.

We emphasize that just adding noise to the output is neither necessary nor sufficient for achieving a meaningful level of privacy. *Noiseless privacy* has been explored in the work of Bhaskar et al., which taps into uncertainty that the adversary may have about the input into the database [BBG+11]; Blocki et al. describe a randomized functionality (the Johnson-Lindenstrauss transform) that offers a relaxation of differential privacy without any additional noise [BBDS12].

More importantly, some (or even substantial) amount of additive noise is not by itself a guarantee of differential privacy. For instance, several of the "doughnut-shaped" noise distributions proposed for disclosure limitation of microdata collected by the US Census Bureau [EZS98] do not achieve differential privacy despite introducing a non-trivial amount of noise.

Consider a different example, which captures the essence of our attack. A counting query returns the number of records in a database satisfying a certain predicate. If the noise added to the accurate output of the query is always even, then the adversary can trivially win the differentially private game (guessing whether the query was evaluated on D or an adjacent D', such that their counts differ by 1) by looking at the parity of the output. In this example, the adversary's probability of success is independent of the absolute value of the noise, i.e., accuracy of the answer to the query.

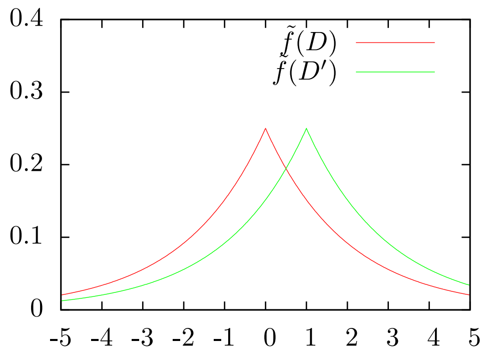


Figure 1: Distributions of $\tilde{f}_\epsilon(D) = f(D) + \mathrm{Lap}(\Delta/\epsilon)$ over two inputs: $f(D) = 0$, $f(D') = 1$, sensitivity $\Delta = 1$, privacy parameter $\epsilon = 0.5$.

2.1 Laplacian mechanism

The basic method for achieving differential privacy is called the Laplacian mechanism. The mechanism is additive, i.e., given the function f it approximates its output by computing the function exactly and then adding noise sampled from a specific distribution. The distribution is Laplacian and in order to preserve privacy of function f, the distribution is scaled to f's sensitivity, defined below.

We say that a deterministic real-valued function $f\colon \mathcal{D} \mapsto \mathbb{R}$ has *sensitivity* at most Δ if for all adjacent databases D and D':

$$|f(D) - f(D')| \le \Delta.$$

The (zero-mean) *Laplacian distribution* $\mathrm{Lap}(\lambda)$ is continuous probability distribution defined by its density function

$$h_\lambda(x) \triangleq \frac{1}{2\lambda} \exp\left(-\frac{|x|}{\lambda}\right).$$

The classic result from literature on differential privacy establishes that the following transformation of f, called *Laplacian mechanism*, is ϵ-differentially-private if f has sensitivity Δ:

$$\tilde{f}_\epsilon(D) \triangleq f(D) + Y, \text{ where } Y \leftarrow \mathrm{Lap}(\Delta/\epsilon).$$

Ghosh at el. [GRS09] proved that (a discrete version) of the Laplacian mechanism is optimal and universal, i.e., it is the only mechanism the curator has to implement to support integer-valued functionalities and a large class of loss functions.

The distribution of $\tilde{f}_\epsilon$ on two inputs where $f(D) = 0$ and $f(D') = 1$ is plotted in Figure 1.

3 Overview of Privacy-Preserving Platforms

Many sophisticated differentially private algorithms have been implemented and evaluated on real data, such as privacy-preserving logistic regressions [CM08], the matrix mechanism [LHR+10] or the exponential mechanism with the multiplicative weights update rule (MWEM) [HLM10]. Doing it right, however, is an error-prone and highly non-trivial task, akin to implementing one's own cryptography. Furthermore, custom analyses of privacy-preserving algorithms are expensive and labor-intensive efforts requiring experts in the loop. It limits applications of differential privacy to problems that justify that expense, excluding scenarios that allow exploration of the query space by untrusted or non-expert participants. Examples of these attractive scenarios include datamining, where analysts are restricted to issuing differentially private queries, and a Netflix-style crowdsourcing competition, where participants train their systems on sensitive data via a differentially private on-line interface.

The first system whose ambition was to facilitate adoption of differential privacy in practice by relieving developers from having to certify privacy of their code was PINQ [McS09, McS10]. PINQ (Privacy INtegrated

Queries) is a declarative programming language that guarantees differential privacy as long as the dataset is accessed exclusively through PINQ queries. PINQ's research goal was to identify a set of data transformation and aggregation operators that would be expressive enough for a range of analytical tasks and yet allow for prove-once-run-everything approach. A PINQ prototype is available as a C# LINQ library.

Airavat [RSK+10] is a Hadoop-based MapReduce programming platform whose primary goal is to offer end-to-end security guarantees, while requiring minimal changes to the programming model or execution environment of big data computations. In particular, it allows execution of untrusted mappers by integrating SELinux-style mandatory access control to the Java Virtual Machine, the Hadoop framework, and the distributed file system. In contrast with PINQ, whose guarantees are language-based and whose current implementation treats developers as cooperating entities, Airavat accepts arbitrary Java bytecode as mappers and then uses trusted reducers and system-level isolation mechanisms to ensure compliance with privacy policies.

Fuzz [HPN11] employs a novel type system [RP10] to facilitate static analysis of sensitivity of arbitrary data transformations. It further limits the subset of allowed mappers through the use of a domain-specific programming language, which requires primitives to supply timeout information and enforces constant execution time on basic data manipulation routines.

The most recent of these systems, GUPT [MTS+12], explores a different point in the design space of privacy-preserving computations. It uses the sample-and-aggregate framework due to Nissim et al. [NRS07], which allows execution of an arbitrary client-provided program over a sample of the original dataset. Accuracy is boosted by running the program multiple times over many samples and averaging the outputs, and privacy is enforced by adding Laplacian noise to the average.

PINQ, Airavat, Fuzz, and GUPT, while different in their design goals, philosophy and languages of implementation, all expose instantiations of the Laplacian mechanism, where the numerical output of a computation of bounded sensitivity is released after adding noise sampled from a Laplacian distribution.

4 LSBs as a Side Channel

As we saw in the previous section, the Laplacian mechanism is the staple of general-purpose differentially private systems. These systems are extremely diverse in their implementation details, yet they share variants of the same sampling procedure, which we reproduce below.

4.1 Sampling from Laplacian

The Laplacian distribution $\mathrm{Lap}(\lambda)$, sometimes called the symmetric exponential distribution, can be thought of as the exponential distribution assigned a randomly chosen sign. The exponential distribution is defined by its cumulative distribution function (cdf)

$$F(t) \triangleq \Pr[Y \leq t] = 1 - e^{-t/\lambda}.$$

The most common method of generating Y is based on the generic procedure, called the inverse sampling method. To draw from the probability distribution given the inverse of its cdf, $F^{-1}(\cdot)$, and a source of uniform randomness U from the $[0,1)$ interval, compute the following:

$$Y \leftarrow F^{-1}(U).$$

To see that the cdf of Y is indeed $F(\cdot)$, one can check that:

$$\Pr[Y \leq t] = \Pr[F^{-1}(U) \leq t] = \Pr[U \leq F(t)] = F(t).$$

Computing the inverse cdf $F^{-1}(\cdot)$ happens to be particularly simple, which results in the following procedure for sampling from the exponential distribution with parameter λ:

$$Y \leftarrow F^{-1}(U) = -\lambda \ln(1 - U)$$

(the distribution $1 - U$ can be replaced by the uniform distribution over $(0,1]$). This method appears in standard references and libraries [Knu97, PTVF07].

To transform Y into the Laplacian distribution one chooses sign at random, resulting in the following procedure:

$$Y \leftarrow (2Z-1)\cdot\lambda\ln(U), \tag{1}$$

where Z is an integer-valued variable uniform over $\{0,1\}$, and U is a real-valued variable uniform over $(0,1]$. In practice most random number generators sample from $[0,1)$, but the probabilities of generating the exact zero or one are small enough that they can be ignored. A twist on the method above uses a single draw from the uniform distribution on $(0,1)$ to generate the sign and the absolute value of the Laplacian variable:

$$Y \leftarrow \mathrm{sign}(r)\cdot\lambda\ln(1-2|r|),\ \text{where } r\leftarrow U-0.5. \tag{2}$$

All differentially private systems considered by this paper use one of the two methods (1) and (2) to sample a Laplacian[1].

4.2 Floating-point numbers and notation

The procedure presented in the previous section is simple and efficient, requiring only a single sample from the $(0,1)$ interval. Unfortunately, simplicity of this procedure is deceptive, since once implemented in floating-point arithmetic, its behavior deviates markedly from its mathematical abstraction.

Before we proceed with discussion of implementations of differentially private mechanisms, we review basic facts about floating-point arithmetic and introduce some notation. For a brief introduction to the subject see Goldberg [Gol91]; Muller et al. serves as a current reference [MBdD+10].

All systems considered in this paper work over double-precision floating-point numbers or just doubles, for short. Doubles occupy 64 bits and reserve 1 bit for sign, 11 bits for exponent, and 52 bits of significand or mantissa. The exponent e is represented as an offset from 1023, thus the smallest representable exponent is 2^{-1022} and the largest exponent is 2^{1023} ($e=0$ is used to represent 0 and subnormal numbers, $e=2047$ represents infinity and NaNs, which we will ignore). The first bit of significand has an implicit value of 1, which gives the total precision of 53 significant bits. If the sign is s, the exponent is e, and the significand is $d_1\ldots d_{52}$, the corresponding floating point number is

$$(-1)^s(1.d_1\ldots d_{52})_2\times 2^{e-1023}.$$

The take-away point is that values representable as doubles are spaced non-uniformly throughout the real line. There are exactly 2^{52} representable reals in the interval $[.5,1)$, 2^{52} reals in the interval $[.25,.5)$, etc.

The ubiquitous IEEE floating-point standard [IEE08] requires the results of basic arithmetic operations (addition, subtraction, multiplication, division) to be exactly rounded, i.e., be computed exactly and then rounded to the closest floating-point number. We will use ability to compute over floating-point numbers with maximal precision in Section 5, where we introduce and analyze a sampling procedure.

To distinguish between real numbers and doubles, we denote the latter as $\mathbb{D}$. The literature on floating-point arithmetic has a name for the value of the least significant digit of a floating-point number, called the *unit in the last place*, or ulp for short. For numbers from the set $\mathbb{D}\cap(0,1)$ (i.e., doubles from the $(0,1)$ interval) their ulps vary between 2^{-1022} and 2^{-53}.

To differentiate between exact, mathematical, functions and arithmetic operations, we will use the following notation, common in the literature: $\oplus,\ominus,\otimes,\oslash$ are floating-point analogues of addition, subtraction, multiplication, and division; $\mathrm{LN}(\cdot)$ stands for a floating-point implementation of the natural logarithm function.

4.3 Sampling from the uniform distribution

Given a non-uniform density of doubles, a uniform distribution over $(0,1)$ is not well defined. It turns out that standards usually offer no guidance and various references and libraries approximate the distribution differently (see Table 1 for a summary). For example, the distribution generated by Java's `Random.nextDouble()` is confined to integer multiples of 2^{-53}.

[1]Current version of Airavat has a stub in lieu of its noise generation routine; we instantiate it with SSJ [L'E]—the library used in reporting performance and accuracy results by Roy et al. [RSK+10].

Reference and Library	Uniform from $[0, 1)$
Knuth [Knu97]	multiples of 2^{-53}
“Numerical Recipes” [PTVF07]	multiples of 2^{-64}
C#	multiples of $1/(2^{31} - 1)$
SSJ (Java) [L'E]	multiples of 2^{-32} or 2^{-53}
Python	multiples of 2^{-53}
OCaml	multiples of 2^{-90}

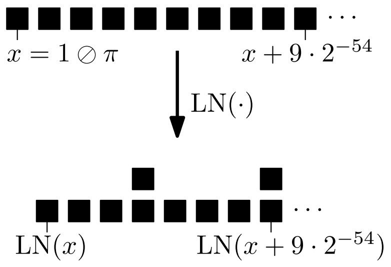
Table 1: Support of random doubles.

Many of these sources of randomness are not cryptographic, i.e., having insufficient entropy and/or predictable, and thus not appropriate for security applications, including differential privacy. We enumerate various strategies for sampling “uniform” floating-point numbers to emphasize lack of consistency in this area.

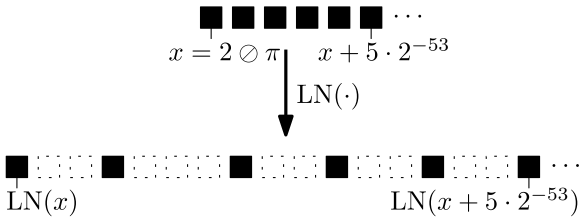
4.4 Sampling from Laplacian: Examples

As we saw in the previous section, samples from the uniform distributions are most likely confined to a small subset of doubles, equally spaced throughout the $(0, 1)$ interval. Even if care is taken to include all doubles in support of the uniform distribution, transforming the uniformly distributed random variable into the Laplacian via formulas (1) or (2) will generate artifacts of its own.

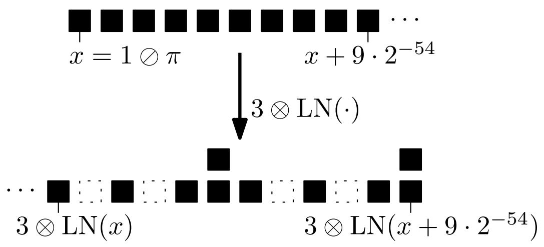
Consider the following, actual example of applying the log function (from C++'s `<math.h>`) to the set of 10 double numbers following $1/\pi$ (there is nothing special about this number from the point of view of sampling Laplacians, which is why it is chosen for illustrative purposes):



It is apparent that some output values are more frequent than others. For some other starting points, another phenomenon is in effect: some potential output values from $\mathbb{D}$ become missing (indicated on the diagram by empty squares):



Multiplying the output of the transformation $U \mapsto -\ln(U)$ by λ, to generate a scaled Laplacian Lap(λ), results in some values being repeated several times and some, none at all. We repeat the experiment with $x = 1 \oslash \pi$ but this time multiply the result by 3, which corresponds to sampling from the Laplacian mechanism with $\Delta = 1$ and $\epsilon = 1/3$:


$x = 1 \oslash \pi$
$x + 9 \cdot 2^{-54}$
$3 \otimes \mathrm{LN}(\cdot)$
$3 \otimes \mathrm{LN}(x)$
$3 \otimes \mathrm{LN}(x + 9 \cdot 2^{-54})$

Since most random number generators do not output all potential doubles (see Table 1), their actual approximation of the Laplacian distribution will have missing values and values that appear more frequently than they should.

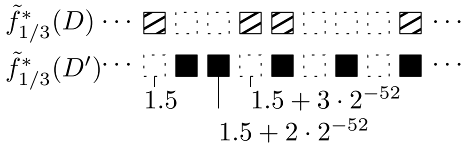
4.5 Attack on floating-point implementation of the Laplacian mechanism

Recall that the reason for introducing Laplacian noise is to convert a deterministic function f into a differentially private one by smoothing its output distribution. The proof of differential privacy of the Laplacian mechanism [DMNS06] depends on the fact that the probabilities of seeing the same output of $\tilde{f}_\epsilon(\cdot)$ on two adjacent inputs are multiplicatively close to each other.

If the floating-point implementation of the Laplacian distribution misses some output values on input D and others on input D', the resulting mechanism cannot satisfy differential privacy. This is exactly what happens when the Laplacian mechanism is instantiated with floating-point arithmetic without appropriate defense mechanisms.

To differentiate between ideal and real-world realizations of the Laplacian distribution and the differentially private mechanism $\tilde{f}_\epsilon(\cdot)$, we will denote floating-point implementations of the Laplacian distribution $\mathrm{Lap}^*_p(\lambda)$ and the corresponding Laplacian mechanism $\tilde{f}^*_{\epsilon,p}(D) = f(D) \oplus \mathrm{Lap}^*_p(\Delta/\epsilon)$, where p is the precision with which the uniform distribution over $(0,1)$ is sampled. If the precision is maximal, i.e., all doubles $\mathbb{D}$ are in the support of the uniform distribution, the p index is dropped.

Let $\Delta = 1$, $f(D) = 0$ and $f(D') = 1$. Consider the distribution of $\tilde{f}^*_{1/3}(D) = 0 \oplus \mathrm{Lap}^*(3)$ (striped squares) and $\tilde{f}^*_{1/3}(D') = 1 \oplus \mathrm{Lap}^*(3)$ (solid squares) around 1.5:



It is apparent from this example that if the output of the Laplacian mechanism is $1.5 + 2 \cdot 2^{-52}$, then its input was D', and if the output is $1.5 + 3 \cdot 2^{-52}$, then the input must have been D.

Observing an output that is infeasible under one distribution, constitutes "smoking gun" evidence that rules out the corresponding input. It gives a simple measure of the distribution's deficiency, or a lower (conservative) bound on the adversary's advantage of breaching differential privacy. Let the support of the distribution be defined as

$$\mathrm{supp}(\tilde{f}^*_\epsilon(D)) \triangleq \{x \in \mathbb{D}\colon\ \Pr[\tilde{f}^*_\epsilon(D) = x] > 0\}.$$

Then the probability of differential privacy guarantee's being broken is at least

$$\Pr[\tilde{f}^*_\epsilon(D) \notin \mathrm{supp}(\tilde{f}^*_\epsilon(D'))],$$

since it captures the probability of the event that a sample from $\tilde{f}^*_\epsilon(D)$ falls outside the support of the distribution $\tilde{f}^*_\epsilon(D')$.

Figure 2 plots this probability, expressed as a function of $\lambda = .01 \ldots 3$, for two floating-point implementations of the Laplacian mechanism: $f^*_{1/\lambda}(\cdot)$ and $f^*_{1/\lambda,53}(\cdot)$, for $\Delta = 1$, $f(D) = 0$ and $f(D') = 1$.

For smaller λ's, which should provide a lower (but still non-trivial) level of differential privacy, the probability that a single sample reveals the input is close to 100%. For larger λ's, presumably yielding stronger differential privacy, the probability of a compromise never becomes less than 35%. Lowering resolution of the

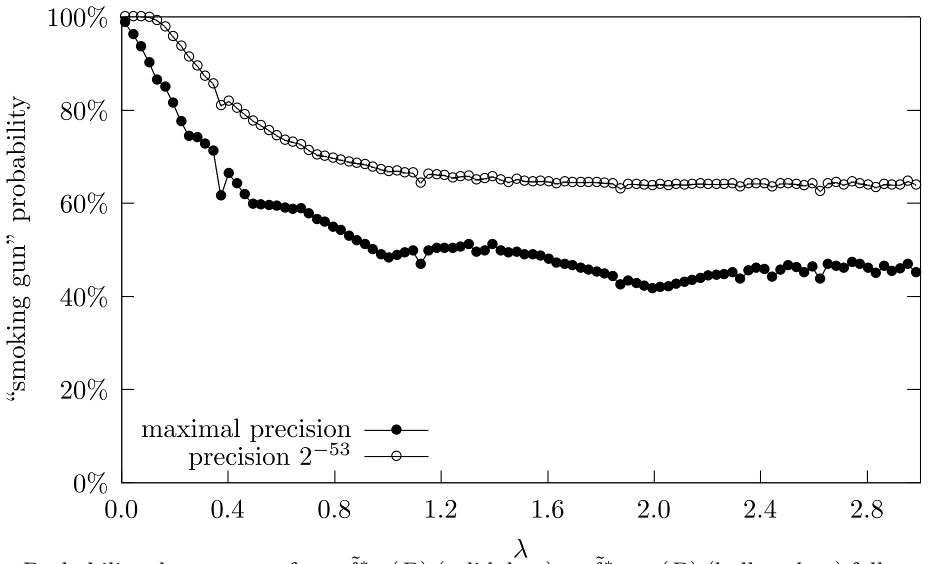


Figure 2: Probability that outputs from $\tilde{f}^*_{1/\lambda}(D)$ (solid dots) or $\tilde{f}^*_{1/\lambda,53}(D)$ (hollow dots) fall outside support of $\tilde{f}^*_{1/\lambda}(D')$ (resp., $\tilde{f}^*_{1/\lambda,53}(D')$), as a function of λ. $f(D) = 0$ and $f(D') = 1$.

uniform distribution (sampling from integer multiples of 2^{-53} instead of $\mathbb{D} \cap (0, 1)$) increases the success rate of the attacker by making the distributions on D and D' more divergent.

This graph, that plots the probability of a catastrophic breach of differential privacy, demonstrates that textbook floating-point implementations of the Laplacian mechanism are insecure. The results are reproducible (with some variations due to compiler options, math libraries, and settings of CPU flags) across languages (C/C++, Java, C#, Python, OCaml), operating systems (Windows, Linux, OS X), and processors (x86, x64). In particular, all four systems surveyed in Section 3 return to the programmer double-precision floating-point answers, and thus enabling this attack.

4.6 Analysis

In addition to empirical data on success probability of the attack, we present heuristic analysis of one particularly important attack scenario, of λ much larger than $f(D)$ or $f(D')$.

We first observe that, somewhat counterintuitively, the attack does not lose its effectiveness for larger values of λ (corresponding to smaller decrements to the privacy budget). Consider the success probability of the attacker after a single interaction with the Laplacian mechanism with $\lambda = 10^6$, $f(D) = 100$ and $f(D') = 101$. The consumed privacy budget is $\epsilon = 10^{-6}$, which would nominally imply a negligible increase in the adversary's confidence ($\exp(10^{-6}) \approx 1 + 10^{-6}$) if its mathematical guarantee held true. Instead, for the textbook implementation of the floating-point Laplacian mechanism, the probability of ruling out one of two adjacent databases is almost 40%. It effectively means that one can consume the privacy budget in arbitrary small increments, without reducing capacity of the side channel. This ability will be exploited in the next section, where we show that the entire database can be extracted by asking sufficiently many adaptively chosen queries.

Another reason to consider the case of a large λ, is that it applies to all values of $f(D)$ and $f(D')$ as long as they are both much smaller than λ.

Figure 3 plots two functions. The first (smooth graph) is the probability density function of $\tilde{f}^*_{1/\lambda}(D)$ for $\lambda = 10^6$ and $f(D) = 100$. The range of the x-axis is $[-2\lambda, 2\lambda]$, which covers $1 - \exp(-2) \approx 0.865$ fraction of output values under that distribution. The second (ragged) line is the probability, computed over small intervals $[a, a + \mu a]$, of the event that an output of $\tilde{f}^*_{1/\lambda}(D)$ in that range is outside the support of the distribution $\tilde{f}^*_{1/\lambda}(D')$:

$$s(a) \triangleq \Pr\Big[x \notin \operatorname{supp}(\tilde{f}^*_{1/\lambda}(D')) \;\Big|\; x \in \operatorname{supp}(\tilde{f}^*_{1/\lambda}(D)) \cap [a, a + \mu a]\Big].$$

We use $\mu = 0.001$ and $f(D') = 101$.

The plot suggests that $s(a)$ (computed over sufficiently small intervals) is smooth except with sharp discontinuities at powers of 2 and certain multiples of λ. Indeed, consider an interval $[2^k \cdot u, 2^k \cdot v]$, when $f(D') + 2^k < \ln 2 \cdot \lambda$, $2^k \gg f(D')$, and $1/2 < u < v < 1$. The set of doubles in this interval is uniformly spaced and has cardinality $2^{53}(v - u)$.

On the other hand, the interval in the range of the uniform distribution that is mapped to $[2^k \cdot u, 2^k \cdot v]$ is approximately (ignoring for a moment the contribution of $f(D) \ll 2^k$):

$$[\exp(-2^k v/\lambda), \exp(-2^k u/\lambda)],$$

which, for $2^k < \ln 2 \cdot \lambda$ is contained in the $[1/2, 1)$ interval. The total number of doubles in that interval is $2^{53} \cdot (\exp(-2^k u/\lambda) - \exp(-2^k v/\lambda))$. The ratio of the doubles in the range and the domain of the mapping $x \mapsto -\lambda \ln(x)$ is thus

$$\frac{2^{53} \cdot (\exp(-2^k u/\lambda) - \exp(-2^k v/\lambda))}{2^{53}(v-u)},$$

which can be approximated for $v \approx u$ as

$$-\exp(-2^k u/\lambda)'(u) = \frac{2^k}{\lambda} \exp(-2^k u/\lambda).$$

As 2^k gets smaller compared to λ, the ratio quickly approaches 0. It means that for smaller values of k, the number of doubles in the image of the mapping is much larger than the number of doubles in its range.

Recall that the doubles output by the uniform distribution are spaced evenly. The logarithm function destroys the arithmetic progression, resulting in a sequence that is not translation invariant. The Laplacian mechanism on inputs D and D' produces two distributions shifted by 1, and we may heuristically assume that a random element from the support of $\tilde{f}^*_{1/\lambda}(D)$ on $[a, b]$ belongs to the support of $\tilde{f}^*_{1/\lambda}(D')$ with probability close to the density of $\tilde{f}^*_{1/\lambda}(D')$ in $[a, b]$, which we estimated earlier. It follows that

$$s(a) \approx \frac{2^k}{\lambda} \exp(-a/\lambda),$$

where 2^k is the smallest power of 2 larger than a. This expression is in remarkable agreement with Figure 3, giving a tight lower bound on the ragged curve in the range $[-(\ln 2)\lambda, (\ln 2)\lambda]$. Although some breaches of differential privacy happen outside that range, our analysis explains almost 90% of all violations for this setting of the parameters.

4.7 A practical attack

To demonstrate that the vulnerability described in the paper is more powerful than a simple breach of differential privacy, we adapt the technique from the previous section to extract the entire content of a database.

Consider a list of alphanumerical records and a query language that reports counts of records sharing a particular prefix. All systems considered by this paper allow this access mechanism.

If histogram counts are protected with Laplacian noise generated with a method vulnerable to our attack, we may reconstruct the entire database by issuing adaptively-chosen queries. We begin by obtaining an upper bound on number of records in the database, then ask a histogram query on the first character of all records, i.e., ask the number of records that start with '\0', '\1',. . . ,'\255'. Initially, all counts up to the upper bound are feasible, but with every query, approximately 40% of the counts not equal to the exact answer can be excluded.

After a unique count for each first character is identified, we append all possible characters to prefixes with non-zero support, and iterate the process until the entire dataset is recovered. Since a single run of the attack is effective even for very large λ according to the analysis of the previous section, the attack can be executed within an arbitrarily small total privacy budget.

We implemented the attack against PINQ, and verified its effectiveness. We were able to reconstruct a database consisting of 18K records in a fewer than 1000 queries with the total privacy budget smaller than 10^{-6}.

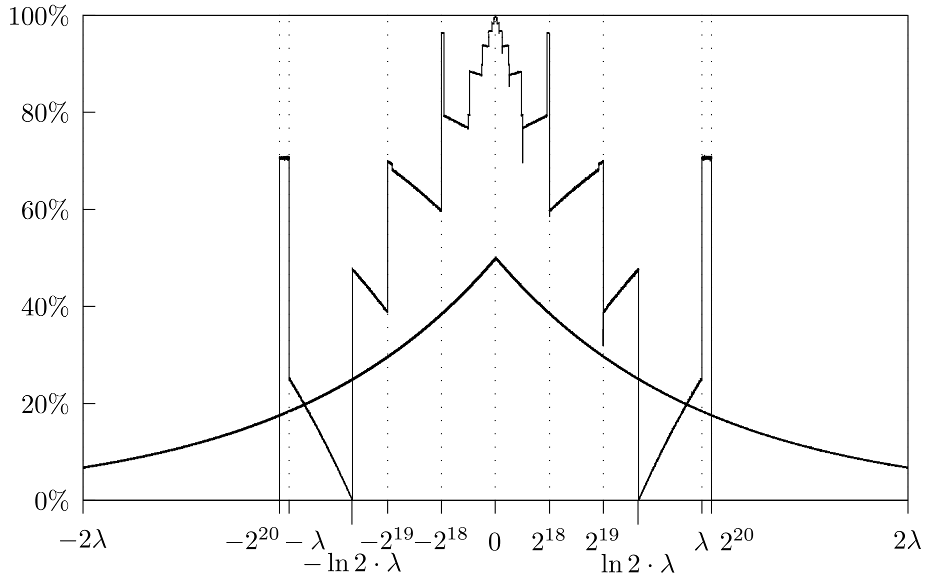


Figure 3: The ragged curve plots the probability for x in $\text{supp}(\tilde{f}^*_{1/\lambda}(D))$ that $x \notin \text{supp}(\tilde{f}^*_{1/\lambda}(D'))$ averaged over small intervals, where $f(D) = 100$, $f(D') = 101$, and $\lambda = 10^6$. The smooth curve is $\text{pdf}(\tilde{f}^*_{1/\lambda}(D))$.

5 Defense

The primary reason why the attack of the previous section succeeds is because the standard floating-point Laplacian sampling procedures (Section 4.1) result in very porous distributions, missing out on many possible output values. That would not be a problem if the missing values were the same for adjacent inputs. Unfortunately, the distributions are not translation invariant, and after being shifted by $f(D)$ and $f(D')$, they diverge significantly.

These effects are observable in vanishingly small scales, bearing almost no relation on numerical accuracy or the number of usable bits of the outputs. Even if a non-private $f(\cdot)$ returns answers with many significant digits, the Laplacian mechanism degrades accuracy of the resulting mechanism, introducing error with the standard deviation of Δ/ϵ.

We leverage both observations in designing the snapping mechanism, whose floating-point implementation is proved differentially private in Section 5.2: the set of output values is made independent of the input, and output values are spaced approximately λ apart. Before we introduce the snapping mechanism, we discuss two approaches that are simple but insecure.

5.1 False starts

An appealing goal for a defense mechanism would be to come up with a method for sampling from a "floating-point aware" Laplacian distribution whose addition guarantees differential privacy even when realized in floating-point arithmetic. One benefit of this strategy is its modularity—it would be sufficient to prove and implement such distribution once, and use it in a variety of contexts that require additive noise.

We analyze two most straightforward approaches towards this goal, and find them ineffective and possibly even more detrimental to privacy than current implementations.

Rounding. It may appear that since gaps in the double-precision outputs of the Laplacian sampling procedure are only noticeable at the smallest possible scale, dropping the least significant bits of the Laplacian could increase security.

Consider, for example, the effect of rounding the output of $\text{Lap}^*(\lambda)$ to the closest integer multiple of, say, 2^{-32}, denoted as $\lfloor \text{Lap}^*(\lambda) \rceil_{2^{-32}}$. It is easy to verify that under some mild assumptions on accuracy of a floating-point implementation of the log function and access to the uniform distribution on $\mathbb{D} \cap (0, 1)$, the resulting distributions for $f(D) + \lfloor \text{Lap}^*(\lambda) \rceil_{2^{-32}}$ and $f(D') + \lfloor \text{Lap}^*(\lambda) \rceil_{2^{-32}}$ will have identical supports if

$f(D) = 0$ and $f(D') = 1$. Moreover, the mechanism confined to these two output values is going to be $1/\lambda$-differentially private.

Recall, however, that the sensitivity of f is an *upper* bound on the difference of f on adjacent inputs. Suppose $f(D) - f(D') = 2^{-33}$, which may easily be arranged by an adversarially chosen function or input. Adding a random variable rounded to a multiple of 2^{-32} will have the effect that the supports of two distributions of the Laplacian mechanism applied to D and D' become completely disjoint, and privacy is going to be lost after a single invocation of the mechanism.

Smoothing. Taking the opposite approach, and trying to smooth the additive Laplacian noise, ensuring that all doubles are in its support, also fails to achieve differential privacy, albeit for a different reason.

Again, consider two potential inputs $f(D) = 0$ and $f(D') = 1$, and additive Lap(1) noise. With probability $\approx 20\%$ the random variable $x = f(D) \oplus \text{Lap}(1)$ belongs to the interval $(0, 1/2)$. Conditional on $x \in (0, 1/2)$, $\text{ulp}(x)$ is strictly less than 2^{-53}. Since we assume Lap(1) takes all possible values, with probability at least 50% x will not be an integer multiple of 2^{-53}. On the other hand, if $1 \oplus y = x$, where y is sampled from Lap(1), then $y \in (-1, -0.5)$, and $\text{ulp}(y) = 2^{-53}$. Since addition is performed exactly in this range, $1 \oplus y$ is going to be an integer multiple of 2^{-53}. It means that in this range the support of $f(D') \oplus \text{Lap}(1)$ will be a proper (and relatively small) subset of the support of $f(D) \oplus \text{Lap}(1)$.

5.2 Snapping Mechanism

In contrast with the first flawed method of the previous section, where the lower bits of the Laplacian noise are simply tossed out, it turns out that doing so *after* adding the noise yields a differentially private mechanism. The technique has been proposed in a different context by Dodis et al. [DLAMV12], where it was shown to be an effective defense strategy when the mechanism is instantiated with *imperfect* randomness, modeled as a Santha-Vazirani source [SV86].

Before we define the mechanism, which is proved to be differentially private in Theorem 1, we introduce some notation. Let U^* be the uniform distribution over $\mathbb{D} \cap (0, 1)$, such that each double number is output with probability proportional to its ulp; S be uniform over $\{-1, +1\}$. $\text{LN}(\cdot)$ denotes a floating-point implementation of the natural logarithm with exact rounding. Function $\text{clamp}_B(x)$ outputs B if $x > B$, $-B$ if $x < -B$ and x otherwise. Λ is the smallest power of 2 (including negative powers) greater than or equal to λ, and $\lfloor \cdot \rceil_\Lambda$ rounds to the closest multiple of Λ in $\mathbb{D}$ with ties resolved towards $+\infty$.

The snapping mechanism, parameterized by B and λ, is defined by the following distribution:

$$\tilde{f}(D) \triangleq \text{clamp}_B(\lfloor \text{clamp}_B(f(D)) \oplus S \otimes \lambda \otimes \text{LN}(U^*) \rceil_\Lambda).$$

Several points are in order. A uniform distribution over $\mathbb{D} \cap (0, 1)$ can be generated by independently sampling an exponent (from the geometric distribution with parameter .5) and a significand (by drawing a uniform string from $\{0, 1\}^{52}$). Efficient algorithms for computing natural logarithm with exact rounding are known and available [dDLM07, CRL]. Rounding to the closest multiple of a power of 2 can be done exactly by manipulating the binary representation of a double. Clamping can be done without introducing additional error.

We additionally assume that the sensitivity of f is 1 (other values can be handled by scaling f in proportion to its sensitivity).

Theorem 1. *The distribution $\tilde{f}(\cdot)$ defined above satisfies $(1/\lambda + 2^{-49}B/\lambda)$-differential privacy when $\lambda < B < 2^{46} \cdot \lambda$.*

Proof. The structure of the proof is as follows. We first define and prove differential privacy of an ideal version of the mechanism $\tilde{f}(\cdot)$, where all computations are performed perfectly with infinite precision and U returns a truly uniform distribution over $(0, 1)$. We then consider for each possible output x of the real mechanism the set of doubles from the support of the U^* distribution that are mapped to x, and prove that its probability mass is well approximated (with a bounded relative error) by the measure of the corresponding set of the ideal mechanism.

The ideal mechanism is defined as

$$\tilde{F}(D) \triangleq \text{clamp}_B(\lfloor \text{clamp}_B(f(D)) \pm \lambda \ln(U) \rceil_\Lambda),$$

where U is uniform over $(0, 1]$ and the sign is chosen from '+' and '-' with equal probabilities.

Since clamp_B is a stable transformation (i.e., if $|x - y| \le 1$, then $|\text{clamp}_B(x) - \text{clamp}_B(y)| \le 1$), applying it to f *before* feeding the result into a differentially private mechanism preserves the mechanism's privacy guarantees. Post-processing the *output* of the mechanism by snapping it to Λ and clamping the result does not affect privacy guarantees either. Therefore, $\tilde{F}(\cdot)$ yields $1/\lambda$-differential privacy, since after the inner and outer operations are removed, what is left is the standard Laplacian mechanism.

Fix the sign in $\tilde{F}(D)$ to be a '+' and similarly fix $S = 1$ in $\tilde{f}(D)$. The case of the negative sign (and $S = -1$) is not fully symmetric due to the direction in which ties are resolved by the rounding operator, but its treatment is analogous to the one below.

Take any x in the support of $\tilde{F}(D)$. Consider the set of U outputs that are mapped by $\tilde{F}(D)$ to x. Since all operations of $\tilde{F}(D)$ are monotone (with the sign fixed), this set forms an interval $[L, R) \subset (0, 1)$. Similarly, let $[l, r) \subset \mathbb{D} \cap (0, 1)$ be the set of U^* values mapped by $\tilde{f}(D)$ to x (when $S = 1$).

We will argue that $|L - R| \approx |l - r|$ in terms of a relative error, which translates into $\tilde{f}(\cdot)$'s guarantee of differential privacy.

We may assume that $f(D) \in [-B, B]$ (otherwise it would be forced to the interval by the inner clamping operation). Since $S = 1$ and the output of $\text{LN}(\cdot)$ is negative, $x \le \lfloor f(D) \rceil_\Lambda$.

Consider the case when $-B < x < \lfloor f(D) \rceil_\Lambda$ (the instances of $x = -B$ and $x = \lfloor f(D) \rceil_\Lambda$ can be treated similarly). In this case, the set of doubles mapped to x by the snapping operator is precisely $[x - \Lambda/2, x + \Lambda/2)$. Denote the endpoints of this interval by a and b.

Under these assumptions, L and R can be computed directly: $L = \exp((a - f(D))/\lambda)$ and $R = \exp((b - f(D))/\lambda)$.

Notice that $0 < L < R < 1$ and, since $b - a = \Lambda$, the ratio $R/L = \exp(\Lambda/\lambda) < \exp(2)$.

The following lemma allows us to bound the relative error of replacing the $[L, R)$ interval with $[l, r)$. We do so by proving upper and lower bounds on r and l.

Lemma 1. *Let $y \in [\Lambda/2, f(D))$. Let $u = \exp((y - f(D))/\lambda)$ and $\eta = 2^{-53}$ (usually called the machine epsilon). Define the following two quantities:*

$$\begin{aligned}
\overline{u} &\triangleq \exp[(y(1+\eta) - f(D))/((1+\eta)^2\lambda)], \\
\underline{u} &\triangleq \exp[(y/(1+\eta) - f(D))(1+\eta)^2/\lambda],
\end{aligned}$$

then

$$f(D) \oplus \lambda \otimes \text{LN}(\underline{u}) \le y \le f(D) \oplus \lambda \otimes \text{LN}(\overline{u}).$$

Lemma 1. This is the only place in the proof of the main theorem where we use the facts that $\oplus$, $\otimes$, and $\text{LN}(\cdot)$ are all implemented with exact rounding. The exact rounding means that the result of an operation is correct within .5ulp of the answer, which translates into the relative error of $1 + 2^{-53} = 1 + \eta$.

The following bounds follow directly from the exact rounding property of arithmetic operations and $\text{LN}(\cdot)$:

$$\begin{aligned}
\text{LN}(\overline{u}) &\ge \ln(\overline{u})(1+\eta) && \text{(since } \ln(\overline{u}) < 0\text{)} \\
\lambda \otimes \text{LN}(\overline{u}) &\ge \lambda \ln(\overline{u})(1+\eta)^2 \\
f(D) \oplus \lambda \otimes \text{LN}(\overline{u}) &\ge (f(D) + \lambda \ln(\overline{u})(1+\eta)^2)/(1+\eta) \\
&\ge y. && \text{(by definition of } \overline{u}\text{)}
\end{aligned}$$

Similarly,

$$\begin{aligned}
\text{LN}(\underline{u}) &\le \ln(\underline{u})/(1+\eta) \\
\lambda \otimes \text{LN}(\underline{u}) &\le \lambda \ln(\underline{u})/(1+\eta)^2 \\
f(D) \oplus \lambda \otimes \text{LN}(\underline{u}) &\le (f(D) + \lambda \ln(\underline{u})/(1+\eta)^2)(1+\eta) \\
&\le y. && \text{(by definition of } \underline{u}\text{)}
\end{aligned}$$

[Lemma 1] □

Analogous bounds can be derived for $y \le -\Lambda/2$.

Lemma 2. *Quantities y, u, $\overline{u}$, $\underline{u}$ are defined as in the setting of Lemma 1. Then*

$$\exp(-3.1B\eta/\lambda)u < \underline{u} < u,$$
$$u < \overline{u} < \exp(2B\eta/\lambda)u.$$

Lemma 2. Consider the ratio $\underline{u}/u$. It can be expressed as

$$\begin{aligned}\underline{u}/u &= \exp\left\{\frac{1}{\lambda}\left[(\frac{y}{1+\eta} - f(D))(1+\eta)^2 - (y - f(D))\right]\right\}\\ &> \exp\left\{\frac{1}{\lambda}\left(-\eta y - 2.1 f(D)\eta\right)\right\}\\ &\geq \exp(-3.1B\eta/\lambda).\end{aligned}$$

We used $(1+\eta)^2 < 1 + 2.1\eta$ and that $f(D), y \leq B$. Similarly,

$$\begin{aligned}\overline{u}/u &= \exp\left\{\frac{1}{\lambda}\left[\frac{y(1+\eta) - f(D)}{(1+\eta)^2} - (y - f(D))\right]\right\}\\ &< \exp\left\{\frac{2f(D)\eta}{\lambda}\right\}\\ &\leq \exp(2B\eta/\lambda).\end{aligned}$$

We used the bound $1/(1+\eta)^2 > 1 - 2\eta$. [Lemma 2] □

To obtain an upper and lower bounds on $|l - r|$ in terms of $|L - R|$, we observe that

$$\underline{L} \leq l \leq \overline{L},$$
$$\underline{R} \leq r \leq \overline{R},$$

by applying Lemma 1 twice. Thus,

$$|\overline{L} - \underline{R}| \leq |l - r| \leq |\underline{L} - \overline{R}|. \tag{3}$$

For this we need $\overline{L} < \underline{R}$, which follows from $R/L = \exp(\Lambda/\lambda)$ and the bounds of Lemma 2.

Applying the multiplicative bounds of Lemma 2, we find that

$$\begin{aligned}\underline{R} - \overline{L} &> \exp(-3.1B\eta/\lambda)R - \exp(2B\eta/\lambda)L && \text{(by Lemma 2)}\\ &= L\cdot(\exp(\Lambda/\lambda - 3.1B\eta/\lambda) - \exp(2B\eta/\lambda))\\ &> L\cdot(\exp(\Lambda/\lambda) - 1)\exp(-7B\eta/\lambda)\\ &= (R - L)\exp(-7B\eta/\lambda), && (4)\end{aligned}$$

and

$$\begin{aligned}\overline{R} - \underline{L} &< \exp(2B\eta/\lambda)R - \exp(-3.1B\eta/\lambda)L && \text{(by Lemma 2)}\\ &= L\cdot(\exp(\Lambda/\lambda + 2B\eta/\lambda) - \exp(-3.1B\eta/\lambda))\\ &< L\cdot(\exp(\Lambda/\lambda) - 1)\exp(5B\eta/\lambda)\\ &= (R - L)\exp(5B\eta/\lambda), && (5)\end{aligned}$$

where both inequalities can be verified given $B\eta/\lambda < 2^{-7}$ and $1 \leq \Lambda/\lambda < 2$.

Together, the bounds on $\underline{R} - \overline{L}$ and $\overline{R} - \underline{L}$, and inequalities (3) imply that $|l - r|$ deviates multiplicatively from $|L - R|$ by at most $\exp(7B\eta/\lambda)$.

According to our definition of U^* (each element of $\mathbb{D} \cap (0, 1)$ is output with probability proportional to its ulp), we observe that

$$r - l = \Pr[U^* \in [l, r)], \tag{6}$$

when $l, r \in \mathbb{D} \cap (0, 1]$.

The only significant difference between the cases of positive and negative signs in the mechanism $\tilde{F}(\cdot)$ (corresponding to $S = +1$ and $S = -1$ in the mechanism $\tilde{f}(\cdot)$), is due to the direction of the rounding operation $\lfloor\cdot\rceil_\Lambda$ in case of ties. It translates in the semiopen interval $(l, r]$ replacing $[l, r)$ from above. The equality (6) has to be modified to account for the case when $\mathrm{ulp}(r) > \mathrm{ulp}(l)$, which is a common occurrence:

$$\begin{aligned} r - l &\le \Pr[U^* \in (l, r]] \\ &= r + \mathrm{ulp}(r) - (l + \mathrm{ulp}(l)) \\ &\le (r - l)(1 + 2\eta), \end{aligned} \tag{7}$$

when $l < r/2$.

The above argument is independent from the actual value of $f(D)$ and applies equally to $f(D')$. Call the corresponding quantities l', r', R', L'.

Finally, putting these bounds together, we have

$$\begin{aligned} &\Pr[\tilde{f}(D) = x] \\ &\quad = \Pr[U^* \in [l, r) \text{ or } U^* \in (l, r]] && \text{(depending on the sign)} \\ &\quad \le (r - l)(1 + 2\eta) && \text{(by (6) or (7))} \\ &\quad \le (L - R)\exp(5B\eta/\lambda + 2\eta) && \text{(by (3) and (by Lemma 2))} \\ &\quad < \Pr[\tilde{F}(D) = x]\exp(5B\eta/\lambda + 2\eta) \\ &\quad < \Pr[\tilde{F}(D') = x]\exp(1/\lambda + 5B\eta/\lambda + 2\eta) && \text{(by differential privacy of } \tilde{F}) \\ &\quad = (L' - R')\exp(1/\lambda + 5B\eta/\lambda + 2\eta) \\ &\quad < (l' - r')\exp(1/\lambda + 12B\eta/\lambda + 2\eta) && \text{(by (3) and (by Lemma 2))} \\ &\quad \le \Pr[U^* \in [l', r') \text{ or } U^* \in (l, r]]\exp(1/\lambda + 12B\eta/\lambda + 2\eta) && \text{(by (6) or (7))} \\ &\quad = \Pr[\tilde{f}(D') = x]\exp(1/\lambda + 12B\eta/\lambda + 2\eta), \end{aligned}$$

as needed for differential privacy of the $\tilde{f}(\cdot)$ mechanism. The expression can be further simplified by plugging in the numerical value of $\eta = 2^{-53}$ and applying $B > \lambda$. □ □

Since the mechanism "snaps" the output of a floating-point computation to the nearest multiple of Λ, the error it introduces relative to the underlying Laplacian mechanisms is at most $\Lambda/2 < \lambda$. Before the outer clamping is applied, the mechanism is unbiased. Depending on the relative magnitude of the range of $f(\cdot)$, λ and B, the bias introduced by clamping may be exceedingly small.

6 Sensitivity Analysis

In this section we briefly describe another vulnerability at the intersection of differential privacy and floating-point arithmetic.

Accurate analysis of sensitivity is essential to most methods in differential privacy. The function f may be a combination of user-provided and trusted codebase, such as Airavat's user-specified mappers and system reducers. For both codebases, subtleties of floating-point arithmetic may dramatically increase sensitivity of the outcome, beyond what mathematical analysis of the underlying mathematical abstraction would suggest.

Consider the following natural example of an aggregation operator: $f(x_1, \ldots, x_n) \triangleq \sum_{i=1}^n x_i$. If the sensitivity of input is 1, i.e., a single user may change at most one input variable x_i by at most 1, then the sensitivity of f is obviously at most 1. However, f's implementation in floating-point arithmetic where summation is performed sequentially in the left-to-right order, denoted by f^*, has much higher sensitivity as demonstrated by the following example:

$$\begin{aligned} &n = 2^{30} + 1, \\ &x_1 = 2^{30}, x_2 = -2^{-23}, \ldots, x_n = -2^{-23}. \end{aligned}$$

Not surprisingly,

$$f^*(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i = 2^{30} - 2^{30} \cdot 2^{-23} = 2^{30} - 128.$$

Increment x_1 by 1. Since $\mathrm{ulp}(2^{30} + 1) = 2^{-22}$, computing $(x_1 + 1) \oplus x_2$ will result in $x_1 + 1$, computing $((x_1 + 1) \oplus x_2) \oplus x_3$ will again produce $x_1 + 1$, etc. In this case

$$f^*(x_1 + 1, x_2, \ldots, x_n) = x_1 + 1 = 2^{30} + 1.$$

The sensitivity of f^*—a floating-point realization of f—on this pair of inputs is 129 rather than 1. This is a manifestation of the accumulated error phenomenon, which can be counteracted by application of the Kahan summation algorithm [Kah65, Hig02]. More complex mechanisms are likely to exhibit similar behavior on adversarially-chosen inputs, complicating the task of validating their privacy claims.

7 Related Work

Modulating the lower order bits of transmitted data is a well-documented covert channel mechanism, whose primary use is to enable communication between processes which are otherwise not allowed to exchange information [Lam73, Gli93].

The main difference between our attack and well-known covert channels is in the intent of the transmitting party. All platforms surveyed in Section 3 accept untrusted code and have to confront the threat of covert communications. An example of a timing attack, to which PINQ and Airavat are vulnerable, and Fuzz and GUPT explicitly address, involves a query that takes inordinate amount of time when it is evaluated on some particular record. Measuring timing of the response leaks information about the input.

The attack described in this paper does not require adversarial code, or any active malicious intent. Any textbook implementation of the floating-point Laplacian mechanism is potentially vulnerable.

Fixed-point or integer-valued algorithms are immune to our attack. Examples of such mechanisms include differentially private computations in a distributed setting [DKM$^+$06], privacy-preserving billing and aggregation [DKR11, KDK11].

The snapping mechanism is closely related to the mechanism proposed by Dodis et al. [DLAMV12], which defends against an adversary who exerts some control over the source of randomness. The adversary in this model is capable of introducing arbitrary bias into the source, as long as each subsequent bit has some constant entropy conditioned on all previously output bits. In particular, the bias may depend on the previous bits and the description of the mechanism. The model is known as the Santha-Vazirani source [SV86], and was shown to imply negative results for existence of certain cryptographic tasks [MP91, DOPS04]. In contrast with Dodis et al., we assume a perfect source of randomness but model inaccuracies due to use of floating-point arithmetic.

8 Conclusions

Floating-point arithmetic is a notoriously leaky abstraction, which is difficult to argue about formally and hard to get right in applications. Non-associativity of basic arithmetic operations, compounding errors, rounding rules, signed zeros, denormals, NaNs, infinities, CPU flags, and hardware bug are complicated enough even in the absence of security concerns.

We initiate study of adapting floating-point algorithms to applications in differential privacy. We describe a practical, efficient attack on a textbook implementation of the Laplacian mechanism that underlies all existing platforms for general purpose differentially private computations.

We describe and prove a post-processing snapping procedure, which does not perceptible increase the error introduced by the Laplacian mechanism. The snapping mechanism is also known to preserve differential privacy when instantiated with a class of weak sources of randomness.

In conclusion we observe that floating-point arithmetic presents a unique security challenge to developers of real-world applications. It is different from convenient mathematical abstractions in ways that are, on the one hand, complex and riddled with many corner- and special cases. On the other hand, its common implementations are standard, predictable, and deterministic, so that these problematic cases can be forced and exploited by the adversary.

9 Acknowledgments

The author thanks Yevgeniy Dodis, Frank McSherry, Benjamin Pierce, and Indrajit Roy for their support and valuable comments.

References

[BBDS12] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The Johnson-Lindenstrauss transform itself preserves differential privacy. In *53rd Symposium on Foundations of Computer Science (FOCS 2012)*. IEEE Computer Society, 2012. To appear.

[BBG$^+$11] Raghav Bhaskar, Abhishek Bhowmick, Vipul Goyal, Srivatsan Laxman, and Abhradeep Thakurta. Noiseless database privacy. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology—ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 215–232. Springer, 2011.

[CM08] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 21 (NIPS)*, pages 289–296. Curran Associates, Inc., 2008.

[CRL] Correctly rounded mathematical library. `http://lipforge.ens-lyon.fr/www/crlibm/`.

[dDLM07] Florent de Dinechin, Christoph Quirin Lauter, and Jean-Michel Muller. Fast and correctly rounded logarithms in double-precision. *Theoretical Informatics and Applications*, 41(1):85–102, 2007.

[DKM$^+$06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *Advances in Cryptology—EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006.

[DKR11] George Danezis, Markulf Kohlweiss, and Alfredo Rial. Differentially private billing with rebates. In *Proceedings of the 13th international conference on Information hiding*, IH'11, pages 148–162, Berlin, Heidelberg, 2011. Springer-Verlag.

[DLAMV12] Yevgeniy Dodis, Adriana López-Alt, Ilya Mironov, and Salil Vadhan. Differential privacy with imperfect randomness. In *Advances in Cryptology—CRYPTO 2012*, 2012. To appear. Full version `http://eprint.iacr.org/2012/435`.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography Conference—TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[DN04] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In Matthew K. Franklin, editor, *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 528–544. Springer, 2004.

[DOPS04] Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (im)possibility of cryptography with imperfect randomness. In *FOCS '04*, pages 196–205. IEEE Computer Society, 2004.

[Dwo06] Cynthia Dwork. Differential privacy. Invited talk. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming—ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[Dwo11] Cynthia Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.

[EZS98] Timothy Evans, Laura Zayatz, and John Slanta. Using noise for disclosure limitation of establishment tabular data. *J. of Official Statistics*, 14(4):537–551, 1998.

[Gli93] Virgil D. Gligor. A guide to understanding covert channel analysis of trusted systems. Technical Report NCSC-TG-030, National Computer Security Center, November 1993.

[Gol91] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, March 1991.

[GRS09] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 351–360. ACM, 2009.

[Hig02] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM: Society for Industrial and Applied Mathematics, 2nd edition, August 2002.

[HLM10] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. *CoRR*, abs/1012.4763, 2010.

[HPN11] Andreas Haeberlen, Benjamin C. Pierce, and Arjun Narayan. Differential privacy under fire. In *USENIX Security Symposium.* USENIX Association, 2011.

[IEE08] IEEE standard for floating-point arithmetic. Technical report, Microprocessor Standards Committee of the IEEE Computer Society, 3 Park Avenue, New York, NY 10016-5997, USA, August 2008.

[Kah65] W. Kahan. Pracniques: further remarks on reducing truncation errors. *Commun. ACM*, 8(1):40, January 1965.

[KDK11] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In Simone Fischer-Hübner and Nicholas Hopper, editors, *PETS*, volume 6794 of *Lecture Notes in Computer Science*, pages 175–191. Springer, 2011.

[Knu97] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming.* Addison-Wesley, third edition, 1997.

[Lam73] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, October 1973.

[L'E] Pierre L'Ecuyer. SSJ: Stochastic simulation in Java. `http://www.iro.umontreal.ca/~simardr/ssj/indexe.html`.

[LHR$^+$10] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. Optimizing linear counting queries under differential privacy. In Jan Paredaens and Dirk Van Gucht, editors, *Symposium on Principles of Database Systems, PODS 2010*, pages 123–134. ACM, 2010.

[MBdD$^+$10] Jean-Michel Muller, Nicolas Brisebarre, Florent de Dinechin, Claude-Pierre Jeannerod, Vincent Lefèvre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, and Serge Torres. *Handbook of Floating-Point Arithmetic.* Birkhäuser Boston, 2010.

[McS09] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In Ugur Çetintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009*, pages 19–30. ACM, 2009.

[McS10] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53(9):89–97, 2010.

[MKA$^+$08] Ashwin Machanavajjhala, Daniel Kifer, John M. Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In Gustavo Alonso, José A. Blakeley, and Arbee L. P. Chen, editors, *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008*, pages 277–286. IEEE Computer Society, 2008.

[MM09] Frank McSherry and Ilya Mironov. Differentially private recommender systems: Building privacy into the Netflix Prize contenders. In John F. Elder, IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki, editors, *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD 2009*, pages 627–636. ACM, June 2009.

[MM10] Frank McSherry and Ratul Mahajan. Differentially-private network trace analysis. In Shivkumar Kalyanaraman, Venkata N. Padmanabhan, K. K. Ramakrishnan, Rajeev Shorey, and Geoffrey M. Voelker, editors, *SIGCOMM*, pages 123–134. ACM, 2010.

[MP91] James L. McInnes and Benny Pinkas. On the impossibility of private key cryptography with weakly random keys. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology—CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 421–435. Springer, 1991.

[MTS$^+$12] Prashanth Mohan, Abhradeep Guha Thakurta, Elaine Shi, Dawn Song, and David E. Culler. GUPT: Privacy preserving data analysis made easy. In *Proceedings of the 38th SIGMOD international conference on Management of data*, SIGMOD '12, New York, NY, USA, 2012. ACM.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 75–84. ACM, 2007.

[PTVF07] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, New York, NY, USA, third edition, 2007.

[RP10] Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: a calculus for differential privacy. In Paul Hudak and Stephanie Weirich, editors, *Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010*, pages 157–168. ACM, 2010.

[RSK$^+$10] Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and privacy for MapReduce. In *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2010*, pages 297–312. USENIX Association, 2010.

[SV86] Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.*, 33(1):75–87, 1986.