Verified Differential Privacy for Finite Computers

Arthur Azevedo de Amorim
arthuraa@bu.edu
Boston University
Boston, Massachusetts, USA

Marco Gaboardi
gaboardi@bu.edu
Boston University
Boston, Massachusetts, USA

Vivien Rindisbacher
vrindis@bu.edu
Boston University
Boston, Massachusetts, USA

Abstract

Differential Privacy not only ensures the anonymity of data, but provides a way of rigorously quantifying and proving how private released information is. A drawback of Differential Privacy is that most suggested implementations are formalized using real numbers. This makes Differentially Private algorithms unimplementable on finite machines. A common way to get around this issue is to implement these algorithms using floating point numbers. However, as shown by Ilya Morinov, these naive implementations lead to privacy breaches using an attack on the least significant bits of the floating point numbers. Therefore, finding ways to verify implementations of Differential Privacy is of significant interest to many. Using Coq, we have formalized a version of the Geometric Truncated Mechanism (GTM), first suggest by Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan using fixed precision arithmetic, and verified that the GTM is in fact Differentially Private.

1 Introduction

Differential Privacy is difficult to implement because of its reliance on probability distributions, which are often continuous. A reasonable solution to this issue, sampling from discrete distributions can lead to slight differences between the underlying probability mass function (PMF) and the implemented random sampling routine. Ghosh et al. proposed to solve this issue with the Geometric Truncated Mechanism, which blurs the result of a query using noise drawn from a discrete geometric distribution and then truncates the result. However, to date, an implementation sampling from this distribution has not been verified.

Following the implementation suggested by Victor Balcer and Salil Vadhan, we have formalized a sampling routine for the GTM in Coq, using fixed precision arithmetic. First, we give an overview of the GTM itself. Then, in Section 3 we describe the inverse transform sampling routine used to sample the underlying distribution. Sections 4 and 5 give a brief overview of the tools used to model probability distributions and the implementation of the GTM. In Section 6, we show how picking an arbitrary size for the uniform distribution used in the inverse transform sampling routine has consequences on the privacy guarantee of the GTM. Finally, in Section 7 we show how choosing specific parameters for the GTM preserves the privacy guarantee achieved by the PMF. All of the theorems stated in the following sections have been formalized in Coq.

POPL '23, Jan 15 - 21, 2023, Boston, MA

.

2 Geometric Truncated Mechanism

The Geometric Truncated Mechanism provides a framework for Differential Privacy using rational numbers. The goal of the GTM is to sanitize the result of a query q. Suppose that q is always bounded to some integer interval $[0, n]$ (e.g., q counts how many entries in a database satisfy a given predicate). Using q, the GTM produces a sample from the following probability distribution:

$$gtpmf(q, output) = \begin{cases} 0 & \text{if } output > n \\ \frac{\alpha^q}{1+\alpha} & \text{if } output = 0 \\ \frac{\alpha^{(n-q)}}{1+\alpha} & \text{if } output = n \\ \frac{1-\alpha}{1+\alpha}\alpha^{|output-q|} & \text{otherwise.} \end{cases}$$

The formal definition of Differential Privacy generally involves some mechanism f which takes as input a query result q from a database D_1, and returns $output = f(q)$. The underlying probability distribution used is generally quantified over ϵ which controls the amount of noise added to q and the privacy guaranteed by the mechanism. We can show that the GTM is α Differentially Private. By definition, this means that, for every pair of neighboring results $|q - q'| \leq 1$, we have the following:

$$\alpha \leq \frac{gtpmf(q, output)}{gtpmf(q', output)} \leq \frac{1}{\alpha}$$

Intuitively, if we think of q as the result of a counting query, as explained above, this means that the probabilities of outputs cannot change much when we run the query after modifying the data of one individual.

3 Inverse Transform Sampling

The GTM is implemented by sampling a random variable from the underlying PMF using inverse transform sampling. Inverse transform sampling samples a number p from a uniform distribution over $(0, 1]$ and finds the first member of the cumulative mass function (CMF) greater or equal to p. The result is a possibly identical distribution to the GTM's PMF. Sampling from a continuous uniform distribution has the drawback of truncation for the sampled value p. Thus, we use a discrete uniform distribution of size T, each member with probability $\frac{1}{T}$.

4 Finprob

In order to implement the inverse transform sampling routine, we use the Finprob library[1]. Finprob formalizes finite probability theory using Extructures[2], a library which implements finite maps. A distribution (*Distr*) is defined as a function f operating over some non-empty set X, where $f\ x \geq 0$ for $x \in support\ f$. The function *mkdistr* is used to instantiate a distribution using a sub-proof that the given set X and the corresponding function f satisfy the requirement for *Distr*. The function *mkprob* is used to instantiate a probability distribution, which is a *Distr* whose *mass* is exactly 1. *mass* is defined as the sum of $f\ x$ for $x \in support\ f$.

An important class of probability distribution for our use case is the uniform distribution. This is instantiated using $unif$, which creates a probability distribution using *mkprob* and where f is a function that returns $\frac{1}{T}$. T is the size of the set X. Finally, *sample* provides a mechanism for sampling from a probability distribution. Conceptually, *sample* samples a value from its first argument (a probability distribution) and uses it to compute another distribution.

5 Implementation

To implement the GTM, we use $unif$ and *sample*. First, we generate the support for the discrete uniform distribution of size T. Then, we create a probability distribution with a proof that the support of the distribution is not empty.

```
Definition discrete_uniform_supp : {fset rat} :=
  fset (map (fun x => x / T) (iota 1 T)).
Definition UniformDistribution :=
  unif discrete_uniform_supp_is_not_empty.
```

Finally, we define the GTM itself, using the inverse transform sampling routine described above. The output *dirac* is a deterministic probability function which gives the sanitized result $output$.

```
Definition gtm (q : nat) :=
sample:
u <- (UniformDistribution T size_unif_nonzero);
dirac(find (fun x => u <= x)
  (geometric_truncated_cdf_vector q)).
```

The GTM has support $[0, n]$ when the size of the uniform distribution $T \geq \frac{1+\alpha}{(1-\alpha)\alpha^n}$, the inverse of a value slightly smaller than any probability returned by the PMF. This allows every possible $output \in [0, n]$ to be sampled.

6 α' Differential Privacy

Consider a naive implementation of the GTM outlined above, where T is chosen arbitrarily. In this case, the GTM and the PMF do not necessarily have identical probability distributions. Therefore, we cannot guarantee the following:

$$\alpha \leq \frac{Pr[gtm(q) = output]}{Pr[gtm(q') = output]} \leq \frac{1}{\alpha}$$

To find the correct privacy bound for this naive implementation, we can use a convenient lemma describing the probabilistic distance of the PMF and the GTM.

Theorem 6.1. $\forall output, output \leq n, let\ ideal = gtpmf(q, output)$ *and* $mech = gtm(q, output)$. *We have* $mech \leq ideal \leq mech + \frac{1}{T} \vee ideal \leq mech \leq ideal + \frac{1}{T}$.

Here, $gtm(q, output)$ is the same as $Pr[gtm(q) = output]$. Naturally, this bound has an interesting consequence on the expected privacy of the GTM. In order to find a privacy bound we can guarantee, we need only consider the worst case of the bound above. There may be a situation in which $gtm(q, output) = gtpmf(q, output) + \frac{1}{T}$ and $gtm(q', output) = gtpmf(q', output) - \frac{1}{T}$. Thus, the GTM will be Differentially Private to some degree of

$$\frac{gtpmf(q, output) - \frac{1}{T}}{gtpmf(q', output) + \frac{1}{T}}$$

By inspecting the probabilities for neighboring queries, we can find some constant α', which is Differentially Private for the naively implemented mechanism.

Theorem 6.2. $\forall output, n,\ output \leq n,\ let\ D = \frac{1-\alpha}{1+\alpha}\alpha^n$ *and* $D' = \frac{1-\alpha}{1+\alpha}\alpha^{n-1}$. $\alpha' = \frac{D - T^{-1}}{D' + T^{-1}} \rightarrow \alpha' \leq \frac{gtm(q,output)}{gtm(q',output)} \leq \frac{1}{\alpha'}$.

7 α Differential Privacy

Although the privacy loss of α' Differential Privacy is not drastic, we would still like to achieve the same bound as that of the PMF. We can prevent the privacy degradation quantified by α' Differential Privacy by tweaking the size of the uniform distribution T. Inspecting the probabilities returned by inverse sampling, we can see that they are all of the form $\frac{x}{T}$. Thus, if we ensure that all probabilities of the true GTM are of this form, we can guarantee that they can be represented exactly by inverse sampling. This is stated formally in theorem 7.1.

Theorem 7.1. *Choose some* $D \in N$ *such that* $D > 2$. *Then intialize* $\alpha = \frac{1}{D}$. *Inverse transform sampling with a uniform distribution of size* $T = (D + 1)D^n$, *preserves* α *Differential Privacy.*

8 Conclusion and Future Work

When formalized correctly, the Geometric Truncated Mechanism allows for an implementation of Differential Privacy on a finite machine. However, our implementation cannot be run as a normal probabilistic program. As future work, we

[1] https://github.com/arthuraa/finprob
[2] https://github.com/arthuraa/extructures

plan to verify an executable version of the GTM. One possibility would be to write a C implemtation of the GTM and verify it using the Verified Software Toolchain. Another potential direction would be to verify the constant time implementation of the Geometric Truncated Mechanism, proposed by Balcer et al.

References

[1] Victor Balcer and Salil P. Vadhan. 2019. Differential Privacy on Finite Computers. *J. Priv. Confidentiality* 9, 2 (2019). https://doi.org/10.29012/jpc.679

[2] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. 2012. Universally Utility-maximizing Privacy Mechanisms. *SIAM J. Comput.* 41, 6 (2012), 1673–1693. https://doi.org/10.1137/09076828X

[3] Ilya Mironov. 2012. On significance of the least significant bits for differential privacy. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM, 650–661. https://doi.org/10.1145/2382196.2382264