Digital Personalized Health and Medicine
L.B. Pape-Haugaard et al. (Eds.)

doi:10.3233/SHTI200124

Better Safe than Sorry - Implementing Reliable Health Data Anonymization

Raffael BILD [a,1], Klaus A. KUHN [a] and Fabian PRASSER [b,c]
[a] *University hospital rechts der Isar, Technical University of Munich, Germany*
[b] *Charité – Universitätsmedizin Berlin, Berlin, Germany*
[c] *Berlin Institute of Health (BIH), Berlin, Germany*

Abstract. Modern biomedical research is increasingly data-driven. To create the required big datasets, health data needs to be shared or reused, which often leads to privacy challenges. Data anonymization is an important protection method where data is transformed such that privacy guarantees can be provided according to formal models. For applications in practice, anonymization methods need to be integrated into scalable and reliable tools. In this work, we tackle the problem of achieving reliability. Privacy models often involve mathematical definitions using real numbers which are typically approximated using floating-point numbers when implemented as software. We study the effect on the privacy guarantees provided and present a reliable computing framework based on fractional and interval arithmetic for improving the reliability of implementations. Extensive evaluations demonstrate that reliable data anonymization is practical and that it can be achieved with minor impacts on executions times and data utility.

Keywords. data protection, anonymization, reliable computing

1. Introduction

Modern data-driven biomedical research, e.g. in the field of precision medicine which tailors healthcare to the characteristics of individuals, increasingly leverages data science methods such as machine learning [1]. However, when creating the required big datasets, laws and regulations mandate stringent privacy protection. Hence, a wide range of safeguards has to be applied, ranging from organizational to technical measures.

Data anonymization is an important technical building block for implementing privacy protection. In this process, data is transformed in such a manner that formal guarantees, e.g. regarding the risk of singling out, linkage or inference, can be provided. Traditional models of privacy protection such as k-anonymity, ℓ-diversity and t-closeness specify syntactic constraints on output datasets, while more recent models like differential privacy formulate requirements for data processing methods [2].

2. Objective

All methods for implementing privacy models require performing changes to data which inevitably leads to a decrease of its utility. To balance a decrease in privacy risks with a

[1] Corresponding Author: Raffael Bild, Institute of Medical Informatics, Statistics and Epidemiology, School of Medicine, Technical University of Munich, Ismaninger Straße 22, 81675 Munich, Germany; E-mail: raffael.bild@tum.de.

decrease of utility, models for quantifying both aspects have been developed. When implementing privacy models in practice, an important challenge lies in the need to reflect their mathematical definitions in software. Privacy models are often formulated over real numbers, which in software are approximated by floating-point numbers with limited precision (typically 64 bits). Computations can therefore yield results that differ significantly from the exact mathematical results [3]. This can make output data of anonymization tools vulnerable to privacy breaches. For example, it has been shown that straightfoward implementations of a common method for achieving differential privacy can be exploited to extract the entire content of a (presumably protected) dataset [4]. However, studies of the effects of floating-point errors on the privacy guarantees provided by other methods for data anonymization are still lacking in the literature.

In this article, we aim to fill this gap, with a focus on investigating further meth-ods which are truthful (i.e. non-perturbative) and hence particularly well-suited for the biomedical domain [5]. For this, we discuss the numerical stability of implementations of various privacy models, including k-anonymity, ℓ-diversity, t-closeness and further methods for achieving differential privacy [2]. Moreover, we present a reliable computing framework, which we have integrated into the open source data anonymization tool ARX [6] to mitigate vulnerabilities resulting from the use of floating-point operations.

3. Methods

3.1. Data Anonymization and Floating-Point Arithmetic

Figure 1 shows an example transformation of an input dataset using a combination of generalization (i.e. the replacement of values with more general, but semantically consistent values), suppression (i.e. the removal of values) and aggregation (i.e. the replacement of values with an aggregate, such as their mean). The example output dataset satisfies 2-anonymity, which means that each combination of attribute values appears at least twice (see [2] for further details). Whether or not k-anonymity is satisfied is easy to determine by simply counting the size of groups of indistinguishable records. Implementing other privacy models, such as ℓ-diversity, t-closeness or differential privacy, requires evaluating mathematical expressions over real numbers, though (cf. Section 3.2).

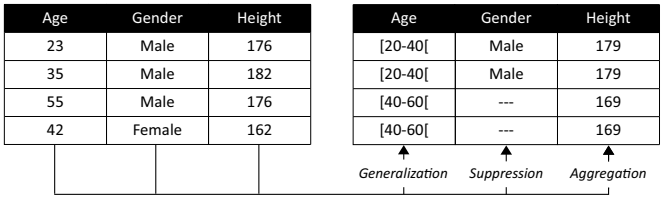

Figure 1. Example of input data and transformed output data.

In computers, real numbers are typically approximated using floating-point numbers. The number of floating-point values which can be represented with a fixed number of bits (typically 64) is finite. Hence, there exists an infinite number of unrepresentable real numbers. Most implementations of floating-point arithmetic adopt the IEEE standard 754 [7]. It specifies that all floating-point operations have to be performed as if it was possible to perform the corresponding operation with infinite precision, and then to round the result to a representable number. This inevitably introduces rounding errors which add up during sequences of calculations [3].

3.2. Numerical Stability of Common Privacy Models

Implementing some privacy models supported by ARX, e.g. k-anonymity [2], doesn't require decimal numbers at all. Implementing others requires significant amounts of decimal arithmetic, though. Examples are (1) t-closeness which basically requires that the distribution of sensitive attribute values in a set of indistinguishable data records is not too different from the corresponding distribution in the overall dataset or (2) (entropy) ℓ-diversity which requires the distribution $(p_1, ..., p_m)$ of sensitive attribute values in each group of indistinguishable records to satisfy $-\sum_{i=1}^{m} p_i \ln(p_i) \geq \ln(\ell)$ [2]. However, by studying possible effects of floating-point error propagation using forward analyses (see e.g. [3] for details), we were able to derive upper bounds for the resulting additive exceedances of the privacy parameters of these models. While a detailed presentation of these analyses is beyond the scope of this article, they showed that the resulting privacy violations are negligible in practice for all syntactic privacy models supported by ARX.

Differential privacy is not a property of a dataset, but a property of a data processing method. It basically guarantees that the probability of any possible output of a probabilistic algorithm does not change "by much" if data of one individual is added to or removed from the input dataset. The Laplacian mechanism and the exponential mechanism are important building blocks for implementating differentially private algorithms [8]. In [9], we have presented a process for implementing k-anonymity while fulfilling (ε, δ)-differential privacy. This approach uses random sampling to introduce non-determinism and the exponential mechanism to optimize the utility of output data. Consequently, unlike the majority of differentially private algorithms, it is truthful and therefore well-suited for processing health data [5].

We were able to calculate an upper bound on the rounding error induced by straightforward floating-point implementations of the exponential mechanism. For this, we applied conservative methodologies described e.g. in [3] followed by an extension of the original proof of the privacy guarantees provided [8] which takes rounding errors into account. While a detailed presentation of this analysis is, again, out of the scope of this article, it showed that the additive exceedance of the expected privacy loss ε is negligible, with values of about 10^{-10} or less in practical settings.

However, the implementation of differential privacy in ARX requires complex calculations to determine the sampling fraction β and the parameter k for k-anonymity to guarantee the requested degree of privacy protection. Investigating our floating-point implementation, we found that the deviations of ε were in the order of 10^{-16} using common values of ε (e.g. 0.01, 0.1, 0.5, $\ln(2)$, 0.75, 1, $\ln(3)$, 1.25, 1.5 and 2). The actual values calculated for the parameter δ, however, deviated drastically, as is shown in Table 1.

Table 1. Relative error of δ for $\varepsilon = \ln(2)$ using a floating-point implementation.

Requested value of δ	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-8}	10^{-9}
Error of δ [%]	13.3	11.1	17.2	28.4	10.0	27.9

3.3. Design of ARX's Reliable Computing Framework

To solve this problem, we implemented a framework comprising different computing technologies that are reliable, i.e. offer strict guarantees for the accuracy of the calculated results: *(1)* Arithmetic using exact arbitrary-precision floating-point numbers. This can be used for calculations involving numbers with a finite amount of digits only. *(2)* Using representations as fractions with arbitrarily long integer enumerators and denominators. This approach can be used to perform exact calculations over rational numbers but it can

become very slow. *(3)* Interval arithmetic [3], which dynamically computes strict bounds on the errors of mathematical operations. The basic idea is not to operate on (approximations of) real numbers, but rather on intervals which enclose the respective exact real numbers. Functions operating on such intervals yield intervals which are guaranteed to include the exact result. For example, addition can be performed by calculating

$$[x_1,x_2]+[y_1,y_2]=[x_1+y_1,x_2+y_2].$$

ARX is implemented in Java and arbitrary prevision arithmetic and fraction arithmetic is well supported by common programming libraries. The number of Java libraries for performing interval arithmetic is, however, known to be limited [10]. Hence, we implemented a basic interval arithmetic framework from scratch while focusing on a minimal amount of easily understandable and verifiable code. We implemented various operators, including the basic arithmetic operators. For more complex functions such as *exp*, *pow*, *log* and *sqrt* we invoke the respective implementations for floating-point values pro-vided by the Java standard library which have clearly defined accuracies. We also imple-mented various comparison operators such as $\leq$ which allow for reliable comparisons by returning the result of comparing the upper and lower endpoint of their operands. These operators are guarded by checks which raise an error if the relation of their operands is not decidable, i.e. if the intervals are overlapping.

We used the methods and operators provided by this framework to implement the parameter calculation process for differential privacy reliably so that the actual degree of privacy protection provided can be at most more conservative than specified by the uer.

4. Results

To evaluate the impact of the reliable parameter calculation on the strictness of the derived parameters we have compared it with a straight-forward floating-point implementation using common values of ε ranging from 0.01 to 2 and $\delta = 10^{-i}$ for $i = 1,...,9$.

The differences between the values of β obtained using both implementations were very small with values of about 10^{-16} in all cases. All values obtained for k were identical except for ten configurations using irrational parameters. This is because these numbers have more significant figures than the other values considered, which resulted in higher rounding errors and hence larger intervals during calculations. In these cases, the values of k obtained reliably were (slightly) higher. Using $\varepsilon = \ln(3)$, the values of k computed differed for $\delta = 10^{-5}$ and $\delta = 10^{-6}$ ($k = 63$ vs. $k = 66$ and $k = 78$ vs. $k = 82$, respectively). The results obtained when using $\varepsilon = \ln(2)$ are listed in Table 2. As can be seen, the absolute differences were at most two. Consequently, for decreasing values of δ, which correspond to increasing degrees of privacy protection, the relative differences between the values of k obtained by both implementations tended to become smaller.

Table 2. Values of k derived from various values of δ and $\varepsilon = \ln(2)$.

δ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
Floating-Point	7	18	30	42	54	67	81	93	105
Reliable	8	20	32	44	56	68	81	95	107

In contrast to results obtained using the floating-point implementation (cf. Table 1), the actual values of ε and δ resulting from reliably calculated parameters k and β were at most more conservative than the privacy parameters specified by the user. In particular, increasing k was necessary to mitigate the violations of δ reported in Table 1. At the same time, the impacts on the intensity of data transformations applied and hence the potential

reductions of data utility are negligible when using recommended parameterizations [9].

We also evaluated the execution times of both implementations on a PC with a quad-core 3.1 GHz CPU, Ubuntu Linux and an Oracle JVM. The results are shown in Figure 2. When decreasing both ε and δ (which corresponds to stronger degrees of protection), the relative execution times tended to increase. Using typical values of $\varepsilon \approx 1$ and $\delta \approx 10^{-6}$, the execution time of the reliable implementation was about four times higher than the time used by the floating-point implementation. In all experiments with $\varepsilon \geq 0.1$, the calculation of parameters terminated in less than one second using both implementations. This contains the range of parameters which is practical for the approach (cf. [9]).

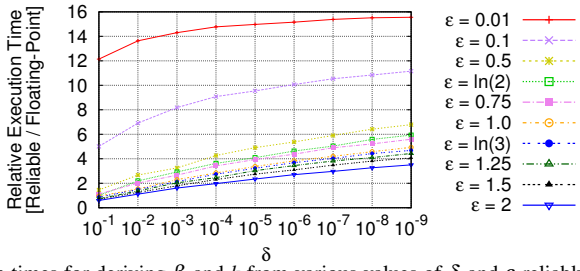

Figure 2. Execution times for deriving β and k from various values of δ and ε reliably relative to the floating-point implementation.

5. Conclusion

In this article, we studied how privacy violations resulting from floating-point implementations of anonymization algorithms can be mitigated. We discussed reliability issues resulting from arithmetic operations for a variety of privacy models including k-anonymity, ℓ-diversity and t-closeness as well as an implementation of differential privacy specifically suited for applications to health data [9]. Moreover, we presented a framework comprising reliable computing techniques, including interval and fractional arithmetic. All results have been integrated into the open source tool ARX. Finally, we examined the impacts of the reliable implementation on output data utility as well as execution times and found both to be negligible in practice when realistic parameters are being used.

References

[1] V. Gligorijević et al., Integrative methods for analyzing big data in precision medicine, *Proteomics* **16** (2016), 741–758.

[2] B. C. Fung et al., *Introduction to privacy-preserving data publishing: Concepts and techniques*, CRC Press, 2010.

[3] N. J. Higham, *Accuracy and stability of numerical algorithms*, vol. 80, Siam, 2002.

[4] I. Mironov, On significance of the least significant bits for differential privacy, *Proceedings of the 2012 ACM conference on computer and communications security*, ACM, 2012, 650–661.

[5] F. K. Dankar and K. El Emam, Practicing differential privacy in health care: A review., *Transactions on data privacy* **6** (2013), 35–67.

[6] F. Prasser and F. Kohlmayer, Putting statistical disclosure control into practice: The ARX data anonymization tool, *Medical data privacy handbook*, Springer, 2015, 111–148.

[7] IEEE Standards Committee et al., 754-2008 IEEE standard for floating-point arithmetic, *IEEE computer society std* **2008**.

[8] C. Dwork, A. Roth et al., The algorithmic foundations of differential privacy, *Foundations and trends® in theoretical computer science* **9** (2014), 211–407.

[9] R. Bild, K. A. Kuhn and F. Prasser, Safepub: A truthful data anonymization algorithm with strong privacy guarantees, *Proceedings on privacy enhancing technologies* **2018** (2018), 67–87.

[10] J. W. von Gudenberg, OOP and interval arithmetic–Language support and libraries, *Numerical software with result verification*, Springer, 2004, 1–14.