PSynDB: Accurate and Accessible Private Data Generation

Zhiqi Huang[†], Ryan McKenna[†], George Bissias [†]
Gerome Miklau[†], Michael Hay[‡], Ashwin Machanavajjhala[*]
[†] Univ. of Massachusetts, Amherst, College of Information and Computing Sciences
[‡] Colgate University, Dept. of Computer Science
[*] Duke University, Dept. of Computer Science
{zhiqihuang, rmckenna, gbiss, miklau}@cs.umass.edu mhay@colgate.edu ashwin@cs.duke.edu

ABSTRACT

Across many application domains, trusted parties who collect sensitive information need mechanisms to safely disseminate data. A favored approach is to generate *synthetic data*: a dataset similar to the original, hopefully retaining its statistical features, but one that does not reveal the private information of contributors to the data.

We present PSynDB, a web-based synthetic table generator that is built on recent privacy technologies [10, 11, 15]. PSynDB satisfies the formal guarantee of differential privacy and generates synthetic tables with high accuracy for tasks that the user specifies as important. PSynDB allows users to browse expected error rates before running the mechanism, a useful feature for making important policy decisions, such as setting the privacy loss budget. When the user has finished configuration, the tool outputs a data synthesis program that can be ported to a trusted environment. There it can be safely executed on the private data to produce the private synthetic dataset for broad dissemination.

PVLDB Reference Format:
Zhiqi Huang, Ryan McKenna, George Bissias, Gerome Miklau, Michael Hay, Ashwin Machanavajjhala. PSynDB: Accurate and Accessible Private Data Generation. *PVLDB*, 12(12): 1918-1921, 2019.
DOI: https://doi.org/10.14778/3352063.3352099

1. INTRODUCTION

In many domains, privacy concerns are a barrier to unlocking the benefits of data analytics and data science. Formal privacy methods based on differential privacy [6, 7] are gaining adoption, by both industry [1, 8] and government agencies [3, 12], but privacy technology remains difficult to deploy and adapt to new settings. This demonstration allows novice users to generate differentially private synthetic data through an easy-to-use web interface and, in addition, provides a path to more sophisticated generation routines through code creation linked to an open-source programming framework, Ektelo [15].

Proceedings of the VLDB Endowment, Vol. 12, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3352063.3352099

Differential privacy supports various modes of use. Differentially private query interfaces can be *interactive*, in which users pose queries one-at-a-time and receive noisy answers. However, each query must consume a portion of their finite privacy loss budget, and they may eventually be locked out of the interface. Queries can be answered privately in *batch mode*, which avoids this problem if queries are known ahead of time. A final mode of operation is synthetic data generation, in which a synthetic "noisy table", conforming to the schema of the original data, is privately produced and given to the user.

Private synthetic data is extremely appealing to users because the output table can be queried freely or used as input to any tools or systems that would use the original data.

But there are fundamental limits on the degree of accuracy that any formally private method can provide. The law of information reconstruction [5] says, informally, that if the method allows "too many" queries to be answered "too accurately" then the private input can be almost completely reconstructed, in which case no formal privacy guarantee can hold. It follows that differentially private synthetic data can only be accurate for a limited class of queries or tasks, not all queries at once. And therefore, it should be generated in a manner that is customized for a specific class of target uses, for which it will provide high accuracy.

While there has been significant prior work on synthetic data generation, existing methods have a number of limitations. Work in the statistics community has produced techniques that are efficient and accurate, but lack rigorous formal privacy guarantees [14]. The problem has been investigated theoretically, however resulting approaches are impractical, requiring exponential runtime [2]. Some more practical approaches have been developed that do satisfy formal privacy guarantees, but cannot be tuned or adapted to prioritize accuracy on the queries/tasks of interest [13, 16].

In this demonstration, we present PSynDB, a web-based tool that allows novice users to synthesize a differentially private table. The interactive tool solicits (non-sensitive) schema information from the user about their private data and then allows the user to build a *workload* of target queries for which the resulting synthetic data should be accurate.

PSynDB is built using recently-developed privacy technologies. Its implementation uses Ektelo [15], a programming framework for differentially private algorithms. In addition, synthetic data is produced using a combination of the High-Dimensional Matrix Mechanism (HDMM) [10] and graphical-model based estimation (PGM) [11].

A web-based tool is easy to use for novice users: it requires no installation of software, no system requirements, or coding. But a downside is that it may be impractical for users to upload their sensitive data to a tool running outside their institution. To address this issue, PSynDB offers two novel features. First, we present the user with visualizations of the expected error rates (on the workload queries) which do not rely on the input data. They can view these error relationships and redesign or refine their workload, without uploading any data. Second, instead of producing a synthetic data set, PSynDB can be used to output Ektelo code, which the user can execute locally on their data without uploading it to an outside server. Thus, users can still enjoy the ease-of-use of the web-based tool without the threat of exposing their data.

2. BACKGROUND & SUPPORTING WORK

PSynDB builds on recent developments in differentially private algorithm design.

Ektelo

Ektelo [15] is a programming framework that supports the implementation of provably private programs. It offers a clean separation between privacy-critical code and other supporting methods, encourages reuse of common subroutines across algorithm implementations, and is expressive enough to capture a wide range of recent privacy algorithms. Ektelo is open-source and we use it both as a reliable backend for the implementation of PSynDB as well as to allow users to move the privacy code to their data after setup has been done using the PSynDB web interface.

HDMM

The high-dimensional matrix mechanism (HDMM) [10] is a differentially private mechanism for releasing answers to a *workload of predicate counting queries*, inspired by the matrix mechanism [9]. Predicate counting queries have the form `SELECT Count(*) FROM R WHERE` ϕ, where ϕ is any boolean formula over the attributes in `R`. Workloads of such queries are quite versatile, expressing histograms, multi-dimensional range queries, data cubes, marginals, or arbitrary combinations thereof.

HDMM works by compiling the workload queries into a more privacy-efficient form, called the strategy queries. The strategy queries are privately answered using the standard Laplace Mechanism [6], and then an inference step is performed to resolve inconsistencies in the noisy measurements. This inference step outputs a vector representation of the estimated database that approximates the true data with respect to the strategy (and workload) queries.

Graphical models

The goal of HDMM and related approaches is to return answers to the workload queries. To generate synthetic data, we therefore have to extend HDMM. We do that by using a recently developed technique based on graphical models [11]. The input to this algorithm is a collection of noisy measurements (i.e., those taken by HDMM), and the output is a graphical model that approximates the data distribution with respect to those measurements. Synthetic data can be efficiently obtained by sampling from the graphical model. We note that combining HDMM with graphical models has been demonstrated to boost the accuracy of the mechanism on the workload [11].

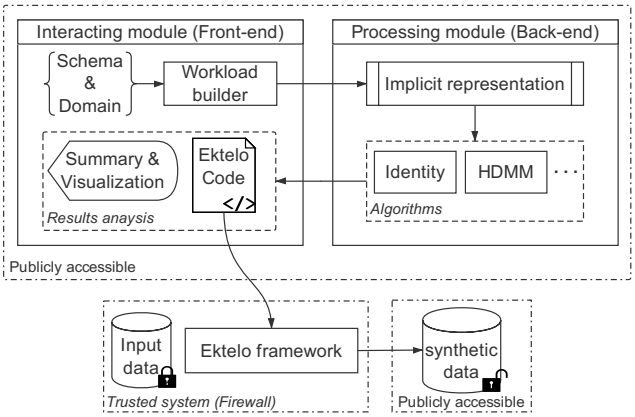

Figure 1: Overview of PSynDB system.

3. THE PSynDB TOOL

Figure 1 presents the system architecture of PSynDB. It has three major components: a front-end interacting module, a back-end processing module, and a local (trusted) Ektelo framework. To generate private synthetic data, PSynDB requires as input a description of the table's schema, including attribute types and domains for each attribute. It provides an interface for the user to define a workload, which consists of a set of predicate counting queries, possibly weighted to reflect their relative importance.

This input is fed to a back-end processing module, which searches for an effective strategy for generating a synthetic table tuned to provide accurate answers to the workload queries. In addition to HDMM, it considers a baseline strategy called Identity, which adds noise directly to a vector representation of the data. A common feature of these strategies is that it is possible to derive a closed-form estimate of the error on the workload queries without actually running the mechanism on the sensitive data. The estimated performance is displayed by the front-end module, which allows the user to interactively explore the expected performance.

The user can iteratively revise the workload structure and observe the effect on performance. When the user is ready to generate synthetic data, the front-end allows them to download a short Ektelo program, which encodes the attribute domain information, a logical representation of the query workload, and the commands to construct synthetic data. This program can then be executed on the sensitive dataset in a trusted environment to produce the synthetic data.

4. DEMONSTRATION OVERVIEW

The PSynDB website guides users through an appropriate definition of the input domain, which is critical for sound privacy semantics, and allows them to define the accuracy criteria by specifying workloads of queries. Given these definitions by the user, PSynDB automatically devises an optimized mechanism, and allows them to browse and inspect the expected error they will receive across the queries in the workload. The process completes before running the mechanism and consuming the privacy budget on the original

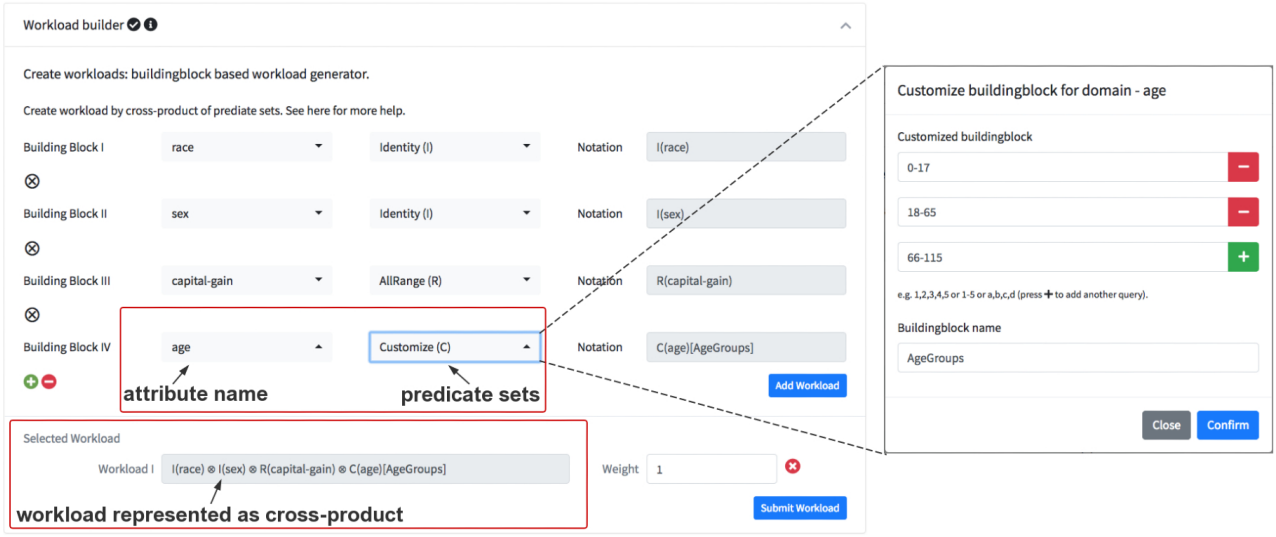

Figure 2: Workload builder in PSynDB.

data. With the workload summarization and error visualization, the user may choose to fine-tune their selections, or re-define their performance criteria before final execution of the mechanism, which will synthesize a differentially private table.

We will demonstrate PSynDB on the well-known Adult dataset [4], which reports demographic and income information for a sample of individuals. We describe the detailed interactive process below in six steps:

Step 1 (Define schema): Initially, the user defines the schema of the table either by manual data entry or by uploading a small sample of data (in `csv` format), from which schema information can be inferred. The user will be able to view and modify attribute domains in the next step.

Step 2 (Define domains): In this step, the user defines attribute domains necessary to specify the workload queries, e.g., if the workload is a histogram of capital gains for each combination of age, race and sex, the user needs to define attribute domains of $\{age, capital\text{-}gain, race, sex\}$ for the Adult dataset. Numerical attributes will be discretized, in which case the user must specify lower/upper bounds and a bucket size. For a categorical attribute, the domain definition contains all possible values and whether it is ordered.

Step 3 (Define workloads): Once the user completes the definition of the schema and attribute domains, she can build her workload. PSynDB supports workloads containing the expressive class of *counting queries with conjunctive conditions*, i.e., the subset of predicate counting queries where the predicates are conjunctively combined. An example of such a query is: `SELECT Count(*) FROM Adult WHERE sex='male' AND race='Black' AND (capital-gain BETWEEN 40000 AND 50000)`.

We provide useful abstractions so that the user may quickly build interesting workloads without having to write down the queries one-at-a-time. In particular, the workloads can be constructed from four common building blocks over a single attribute A: Identity, Prefix, AllRange, Total. Each building block contains a set of predicates, given below (where $dom(A)$ denotes the domain of attribute A):

1. Identity (I): $\phi_i = (A = a_i)$, $\forall a_i \in dom(A)$.
2. Prefix (P) $\phi_i = (A \geq a_i)$, $\forall a_i \in dom(A)$
3. AllRange (R): $\phi_{ij} = (a_i \leq A \leq a_j)$, $\forall a_i, a_j \in dom(A)$
4. Total (T): $\phi = \text{True}$

A workload can consist of a collection of attributes, and a corresponding building block for each attribute. The workload encodes the cartesian product of predicates in the building blocks, where individual queries from each building block are conjunctively combined. For example the set of queries that report all ranges of capital gains for each combination of sex and race could be encoded as $\text{Identity}(Sex) \times \text{Identity}(Race) \times \text{AllRange}(Capital\text{-}gain)$. Omitted attributes have an implied Total predicate.

The Identity building block is useful for categorical attributes with unordered domains, whereas Prefix and AllRange make sense for attributes with ordered domains, such as those arising from discretizing a numeric attribute. These building blocks can encode a wide variety of interesting workloads. To capture even more general workloads, we have a fifth custom building block allowing the user to express arbitrary predicates on a single attribute, e.g., to build counting queries for different *age* groups like $\{0-17, 18-65, 66-115\}$, The system uses C (Customize) to represent the user defined predicate sets. Such customization, along with the predefined predicates, allows the flexibility to describe a wide variety of query workloads.

Figure 2 shows the interface in PSynDB for an example consisting of a single specified workload. Moreover, the tool allows the user to construct multiple such workloads, which can then be combined using a union operation (not shown in Figure 2).

Step 4 (Run optimization): When workloads are submitted to the back-end processing module, PSynDB runs the HDMM optimization to find a good query strategy for the given workload, and for comparison purposes, also considers the baseline Identity strategy. Then it calculates the expected error for every query in the workload under these strategies. Because HDMM represents the workload implicitly using Kronecker products, PSynDB can handle large domains and workloads. However, when the number of queries in the workload is too large to enumerate explicitly, we instead calculate the expected error for a uniform sample of one million workload queries.

Workload	Number of Query	Method & Description	Root MSE
I(race)⊗I(sex)⊗R(capital-gain) ⊗C(age)[AgeGroups]	278154	Identity with Laplace noise (Identity)	51.5278
		High-dimensional Matrix Mechanism (HDMM)	29.8245

Table1: Compare different methods for single workload

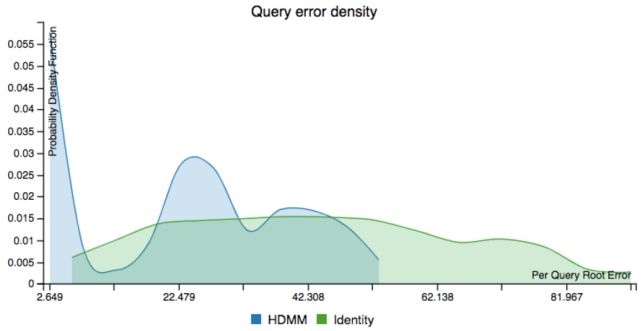

Figure 3: Query error overview for a given workload

Step 5 (Visualization): In the front-end module, the user is now able to view the expected error for the workload and how the errors are distributed across queries. This gives the user the information they need to fine-tune their workload and determine an appropriate privacy budget before running the mechanism on their sensitive data.

Figure 3 shows the error distribution on the example workload for the optimized query strategy (HDMM) as well as for a baseline query strategy (IDENTITY). It shows that workload query error is considerably lower using HDMM. The table above the plot reports the expected error averaged across all queries in the workload for two mechanisms.

When the workload is the union of two or more subworkloads, we also visualize the error distribution for each subworkload. This is shown in Figure 4, where two distributions are plotted, one for the first subworkload and one for the second subworkload. The table above the figure shows the expected error for each subworkload, averaged across the queries in that subworkload.

The user can utilize these visualizations to refine her design of the workload. For example, the visualizations may reveal outliers that could be avoided by re-weighting the workload. Similarly, after viewing the visualizations, the user may decide to add or remove queries to the workload. This process can be iterated until the user is happy with the expected errors of the mechanism.

Step 6 (Generate output): If the user is ready to synthesize data in a trusted environment, PSYNDB can generate Ektelo code which includes schema definitions, attribute domains, workload submission, and the best (lowest error) query strategy in a file. By downloading the file, the user can locally execute Ektelo code, using her sensitive data as input, to generate a synthetic dataset.

Acknowledgements. This work was supported by the National Science Foundation under grants 1408982, 1409125, 1421325, and 1409143; and by DARPA and SPAWAR under contract N66001-15-C-4067. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

ID	Workload	Number of Query	Root MSE
workload I	I(race)⊗I(sex)⊗R(capital-gain)⊗C(age)[AgeGroups]	278154	29.8382
workload II	I(race)⊗I(sex)⊗R(capital-gain)⊗P(age)	10755288	27.6874

Table1: HDMM1 - Evalute each workload separately

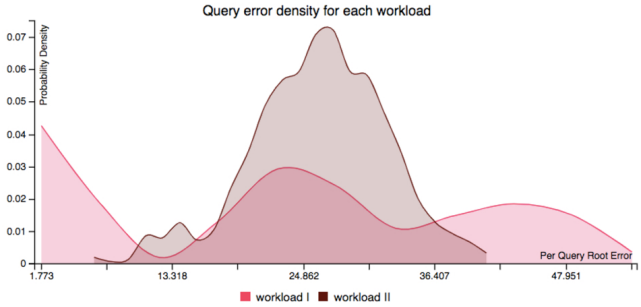

Figure 4: HDMM error overview for multi-workloads

5. REFERENCES

[1] Apple Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 2017.

[2] A. Blum, K. Ligett, and A. Roth. A learning theory approach to noninteractive database privacy. *Journal of the ACM*, 60(2):12, 2013.

[3] 2010 Census Summary File 1, Census of Population and Housing, 2012.

[4] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017.

[5] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Principles of Database Systems*. ACM, 2003.

[6] C. Dwork, F. M. K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. 2006.

[7] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Found. and Trends in Theoretical Computer Science, 2014.

[8] Ú. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067, 2014.

[9] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Principles of Database Systems*, pages 123–134. ACM, 2010.

[10] R. McKenna, G. Miklau, M. Hay, and A. Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *PVLDB*, 11(10):1206–1219, 2018.

[11] R. McKenna, D. Sheldon, and G. Miklau. Graphical-model based estimation and inference for differential privacy. In *ICML*, 2019.

[12] OnTheMap Web Tool. `http://onthemap.ces.census.gov/`.

[13] H. Ping, J. Stoyanovich, and B. Howe. Datasynthesizer: Privacy-preserving synthetic datasets. In *Conference on Scientific and Statistical Database Management*, 2017.

[14] J. P. Reiter. Satisfying disclosure restrictions with synthetic data sets. *Journal of Official Statistics*, 18(4):531, 2002.

[15] D. Zhang, R. McKenna, I. Kotsogiannis, M. Hay, A. Machanavajjhala, and G. Miklau. Ektelo: A framework for defining differentially-private computations. In *SIGMOD*, 2018.

[16] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems*, 42(4):25, 2017.