A Platform for Differential Privacy*

Version: June 18, 2020

Summary

Differential privacy is a formal, mathematical conception of privacy preservation. An algorithm is differentially private when it injects a precisely calculated quantity of noise to any statistical query, masking the possible contribution of any one individual to the result. This provides a gold standard definition of privacy protection for data scientists who want to analyze data that contains personal information that must remain private.

The open source project provides several basic building blocks that can be used by people involved with sensitive data, with implementations based on vetted and mature differential privacy research. It aims to connect theoretical solutions from the academic community with practical lessons learned from real-world deployments and to make differential privacy broadly accessible to future deployments. The core library is a native runtime that is built in Rust to be memory safe and fast. It can be used to safely build differentially private releases from native code running on various devices. It has a SQL data access layer that allows users to compose analysis graphs using the SQL language. The data access layer supports a wide variety of SQL database engines. To ease deployment, it includes a sample hosted service that shows users how to compose heterogeneous queries over the same dataset, fronted by a REST-based endpoint.

Introduction: The concept of differential privacy

Common data-sharing techniques that attempt to preserve privacy either bring great privacy risks or great loss of information. Moreover, the increasing ability of big data, ubiquitous sensors, and social media to record lives in detail brings new ethical responsibilities to safeguard privacy. Differential privacy, with roots in cryptography, is a formal, mathematical conception of privacy preservation. It provides a gold standard definition of privacy protection for data scientists—including industry analysts, scientific researchers, and data-driven policy makers—who want to analyze data that contains personal information that must remain private. Differential privacy guarantees that any released statistical result does not reveal information about any one single individual.

It is key to understand that differential privacy is not an algorithm, but a definition. An algorithm is differentially private when it injects a precisely calculated quantity of noise to any statistical query, masking the possible contribution of any one individual to the result. Then it is mathematically provable that no possible combination of queries or model results can tease out information specific to any individual data subject. This is a nontrivial guarantee, because in the absence of sufficient noise—or when differential-privacy theory is not applied—adversaries can combine many aggregated statistics to infer sensitive attributes of specific individuals.

---

* This white paper was written by Sarah Bird (Microsoft), Joshua Allen (Microsoft) and Kathleen Walker (Allovus Design Inc.), based on materials produced by the differential privacy team, OpenDP, and others.

Privacy is related to, but different from, confidentiality. Tools that assure confidentiality, such as encryption, are often a front-line defense for privacy. However, privacy is concerned with anything that can be inferred about individuals—from the initial ingestion of data, through all of the possible downstream data products and summaries, all the way to decisions that are made from these downstream data products. In the typical setting, private data is sequestered to a controlled environment where it can be strictly managed with confidentiality, security, and policy tools. Data products and decisions built on top of this private data may be distributed more widely, creating additional vectors for adversaries and requiring more attention to privacy-preserving defense in depth.

The OSS project

Deploying differential privacy in practice remains heavily challenging: the concepts are new to most data practitioners, the cryptographic expertise required to vet code is rare, and the increasingly sophisticated attacks that can leak private data proliferate. The project aims to connect theoretical solutions from the academic community with practical lessons learned from real-world deployments and to make differential privacy broadly accessible to future deployments.

The platform works as a curator system. Specifically, the project provides several basic building blocks that can be used by people involved with sensitive data, with implementations based on vetted and mature differential privacy research.

The tools are focused primarily on “global model” of differential privacy, as opposed to the “local model.” In the global model of differential privacy, a trusted data collector is presumed to have access to some private data, and wishes to protect public releases of aggregate information. They are designed to be useful in a wide variety of deployment scenarios where the global model applies.

At the core is a small native runtime that can be used to safely build differentially private releases from native code running on various devices. Around that library, the platform provides tools to help with deployment of differential privacy, such as bindings for data scientists to validate and perform differentially private analyses in notebooks and a data access library that allows differentially private analytics.

For this release, the platform focuses on scenarios where the analyst or data scientist is trusted by the data curator—the analyst and the data curator occupy the same trust boundary.
There are instances where the analysts and the data curator may be different entities, and the analysts may be untrusted. In those situations it is important to have security boundaries between the analyst and the code that runs the differential privacy. Although this was not a focus in the current release, the platform does, however, provide some building blocks that can be implemented for this scenario.

The core

The core library is a native runtime that is built in Rust to be memory safe and fast. It includes implementations of the most common mechanisms and statistics, as well as some utility functions like filtering, imputation, and others. It has an interface for composing operations in an analysis graph that

can be validated for privacy properties. This validator is really the key functionality, as it ensures users are providing formal privacy assurances.

The core library includes a runtime that executes the analysis graph over files or in-memory data. The design is open, and the interface is based on protocol buffers, so other implementations can interoperate.

The core library is designed to be pluggable, allowing inclusion of new algorithms, including both the "aggregate and add noise" and "sample and aggregate" approaches. The primary releases available in the library and the mechanisms for generating these releases are shown below.

Statistics	Mechanisms	Utilities
Count	Gaussian	Cast
Histogram	Geometric	Clamping
Mean	Laplace	Digitize
Quantiles		Filter
Sum		Imputation
Variance/Covariance		Transform

For a listing of the extensive set of components available in the library, see this documentation.

The system

On top of the core library, the platform has a SQL data access layer that allows users to compose analysis graphs using the SQL language. This makes it easy to port over existing dashboards and analytics workflows to be differentially private. The SQL language is a limited subset of the operations that can be specified in the full analysis graph.

The SQL data access layer transparently intercepts calls to backend databases that support SQL-92, applying differential privacy before returning results. The library includes support for SQL Server, PostgreSQL, Spark, SQLite, Pandas, Dataverse, and Presto.

While the core library is designed to consume rows of data in-memory or from disk, the SQL layer feeds exact aggregates to the differential privacy algorithms. In cases where data are stored in optimized indices in the database engine, this can substantially speed up processing.

Because this system's SQL support functions as a data access layer, it is straightforward to add support for other backends, such as Oracle or DB2. The platform provides interfaces that can be extended by others, and more databases will be added in the future. However, this approach limits the ability to target SQL backends with algorithms that require row-based access. For this reason, it's expected that proprietary differentially private capabilities will eventually migrate closer to storage engines, and the

platform has been designed with this functionality in mind. (One recent example of differential privacy implemented in an engine, rather than at the data access layer, is Google's differentially private extensions for PostgreSQL.)

To ease deployment, the platform includes a sample hosted service that shows users how to compose heterogeneous queries over the same dataset, fronted by a REST-based endpoint. Users learn how this can be hosted in a cloud environment backed by the open source MLflow execution environment, and how to track combined epsilon across queries that use IBM's differential privacy libraries and a differentially private SGD using Pytorch. This sample service keeps track of cumulative privacy cost only, but could serve as a starting point for something that prevents query after budget has been spent.

Finally, the platform provides a stochastic evaluator tool that can drive any black box privacy algorithm to test for adherence to privacy promises and checks the accuracy and bias. It is helpful to think of this as somewhat like a fuzzer that can try to find issues with new algorithms and somewhat like a robot that can be used to explore the accuracy and privacy properties of a particular setting.

The community and contributions

This differential privacy project is open source software contributed to the OpenDP consortium under the MIT license and maintained according to the OpenDP governance model.

We appreciate all contributions. However, if you plan to contribute new features, utility functions, or extensions to the core, we ask that you first open an issue and discuss the feature with us. This is because we may be taking the core in a different direction than you might be aware of. A pull request sent before this discussion could be rejected.

Please let us know if you encounter a bug when opening an issue. For bug-fixes, we welcome pull requests without prior discussion.

Learn more about OpenDP and the platform on GitHub.

Editors note: A previous version of this post referred to the name of the platform as WhiteNoise