A Privacy-Preserving Validation Server Prototype

Silke Taylor  
Urban Institute

Graham MacDonald  
Urban Institute

Kyle Ueyama  
Urban Institute

Claire Bowen  
Urban Institute

September 2021

The authors welcome feedback on this working paper. Please send all inquiries to staylor@urban.org.

We would like to thank Arnold Ventures, the National Science Foundation National Center for Science and Engineering Statistics, and the Sloan Foundation for their generous support of this work. We’d also like to thank our incredible partners at the IRS Statistics of Income Division, an amazing team of dedicated public servants, as well as the Harvard OpenDP and Microsoft SmartNoise teams for their technical assistance and open source libraries. We are also grateful for our amazing group of testers; Forum One, for the design and front-end build; and our incredible advisory board of key representatives and top experts across industry, academia, and government for their input in this work.

Urban Institute working papers are circulated for discussion and comment. Though they may have been peer reviewed, they have not been formally edited by the Department of Editorial Services and Publications. The views expressed are those of the authors and should not be attributed to the Urban Institute, its trustees, or its funders.

Copyright © September 2021. Silke Taylor, Graham MacDonald, Kyle Ueyama, and Claire Bowen. All rights reserved.
Abstract

This technical white paper provides an overview of the privacy-preserving validation server prototype from project, researcher, and technical perspectives. In the first section, we discuss the purpose of the project, our process for building the prototype, key concepts, and how we envision a researcher might use the tool. In the second section, we provide detail on how this system could be adopted by other organizations. We explore technical details about how the front end, application programming interface, and back end function; how the prototype can be set up and configured; and how the system might be modified and improved.
A Privacy-Preserving Validation Server Prototype

The goal of our project is to produce a prototype privacy-preserving validation server technology system that could be implemented to allow many more researchers to analyze confidential data in the federal government.

Currently, researchers interested in using data analytics to inform the public debate must work as a federal government agency employee, use publicly available data, or apply to and go through a rigorous clearance and review process to analyze the confidential data. Though the public data are extremely useful for certain purposes, such as tax modeling, they are necessarily redacted, blurred, and edited to a degree that makes many important analyses impossible. And though accessing confidential data would be the obvious solution in most cases, the clearance process can take a long time and is often limited to a handful of trusted research.

Our project would allow researchers to apply for access to a tool that, we envision, would not require a lengthy clearance process to analyze confidential data and would therefore significantly reduce the time it takes to produce valuable public insights. The tool would also begin to replace manual review of requests for disclosure of confidential information, which naturally leads to staff time bottlenecks, with automated privacy-preserving noise addition.

Under our system’s current design, researchers can submit analyses and receive results that are slightly altered to automatically preserve privacy. During any part of the research process, that means no researcher would have direct access to view the confidential data in any form.

Initially, statistical disclosure offices at government agencies might wish to manually review many of these submissions. Our hope is that as trust grows, fewer submissions would need to be manually reviewed. The automation of disclosure review could significantly increase the number of valuable projects that contribute to better understanding of our society and, we hope, the creation of more policies based on that research evidence.

As part of this prototype, our system is built using public-use data to further vet the system and allow for review by many parties interested in helping the team further develop and improve the technology. The system is protected by user-based authentication, and it is only accessible by invitation at this time as we test with trusted users to improve the tool’s usefulness.
Our Process for Building the Tool

The Urban team partnered with the digital agency Forum One to create a lean, user-centered design approach to first design and iterate the system with users and then build the initial prototype over nine months. The team initially decided on a priority audience of researchers at all levels in academia or research organizations, whether they be research assistants graduate students, professors, or senior researchers. Then the team conducted an initial wireframe build and iterated that wireframe with multiple feedback sessions from researchers, privacy experts, and technologists at all levels at the Urban Institute.

After landing on a general approach to the design, the Forum One team created a high-fidelity mock-up that allowed users to explore an application that looked like the site without formally building the full tool. The team then moderated tests conducted individually with internal and external researchers. Based on their feedback, the team further improved the tool and created a new version of the mock-up application, which we then tested again in a moderated fashion with a different set of individuals. Considering the final set of feedback and further input from the Urban project team, the Forum One team put together a final design for the product that was ready to be built. Urban’s internal privacy, research, and technology teams were a key part of the process, constantly checking the feedback to determine what was feasible for this prototype and what needed to be delayed for future improvements.

After providing the final mock-up, the Forum One team began working on the front-end website while the Urban team began working on the application programming interface (API) and back end to produce the prototype system we have today.

Throughout the process, the Urban technology team conducted research on security requirements for the system with input from government and Urban security experts and documentation. We also explicitly decided to use open-source technology to allow the system to be flexible and adaptable to future use, as well as to enable accelerated development by parties interested in using and adapting the tool. Given these efforts and feedback from key government stakeholders, we believe the tool can be adapted for future uses of confidential government data.

Key Concepts

**Differential privacy.** This is a strong, formal definition of privacy. In the data privacy and confidentiality community, formally private methods mean researchers can mathematically prove
and quantify the worst-case scenario of privacy being leaked from a public data publication or statistic. Though several different methods can be used to implement differential privacy in practice, they all add noise to data analysis results proportionate to a specified privacy budget and the sensitivity of the data requested. This allows data owners, such as government statistical agencies, to quantify the maximum amount of privacy loss incurred with any release of information.

**Sensitivity.** Differential privacy experts often use “sensitivity” to describe how robust a given data analysis request, or query, is to outliers. Experts quantify this sensitivity by measuring how much the result of an analysis changes in the confidential data given the absence or presence of the most extreme possible record that could be in the data population but might not be observed in the data. In other words, if privacy experts do not know how a bad actor will attack the data, the privacy expert should assume the worst-case scenario: that the attacker has information on every observation of the data but one and has unlimited computational power. This means differential privacy tries to account for any possible version of the data that could exist, protecting against future data releases and new technologies. If the result from an analysis is too sensitive to outliers for any possible version of the data, then a differentially private method will add more noise to protect the records currently present in the data.

Though sensitivity is not visible to the researcher in our tool, it is important for data analysts to understand, because analyses with higher sensitivity require a higher privacy budget to attain the same accuracy level as queries with lower sensitivity. Analyses that attempt to study the average capital gains of the top 10 earners in the country, for example, will need an enormous privacy budget to provide statistics accurate to within $1,000, on average. Analyses that attempt to study the average capital gains of the entire US population, however, will need a very small privacy budget to achieve similar accuracy.

**Privacy budget.** Often described by the Greek letter epsilon in the privacy literature, a privacy budget allows the data owner to better understand and quantitatively limit the information being released. Paired with information on typical errors for an analysis, the privacy budget also allows the researcher or analyst to better understand the trade-offs of their analysis, so they only request the data they need to be publicly released at an appropriate accuracy level. In a sense, a privacy budget is a numerical expression of the desire to achieve a balance between the optimal protection of individual privacy and the public value of data.
**Review and refinement budgets versus public release budgets.** In our prototype, we present two privacy budgets. The first is a review and refinement budget, which we imagine would be fairly generous, but in return, researchers and analysts would be prohibited from removing any of the results from the secure environment. The results returned from any analyses using the review and refinement budget would only be for experimentation, review, and refinement of any final analyses. We believe the review and refinement budget should have a finite cap, despite the fact that no results may be released publicly, to enable data owners to meet stringent government requirements about the confidentiality and security of confidential data.

The public release budget is a separate privacy budget that we imagine would be less generous. However, any analyses using the public release budget would be cleared for full public release. Though managing two budgets adds complexity, we felt the trade-off was valuable enough to users, many of whom rated data experimentation in a secure environment as key to any analysis.

**Public-use data.** For this project, we use the IRS Statistics of Income Public Use File as our example dataset.

**Confidential data.** In the future, we hope the confidential data used by the system will involve true confidential government records. However, for the prototype system, we use the IRS Public Use File as a stand-in for confidential data to allow an array of users to test and improve the system.

**Secure environment.** In the future, we imagine this system could exist within one of two environments, depending on the implementation needs of the data owners. In the first environment, after researchers apply and are approved for access, they would enter a facility, perhaps one established by the National Secure Data Service or the agency holding the data,¹ to perform analyses and request public release of results. The facility would consist of hardened machines and other necessary physical and software safeguards. Another alternative is to allow for remote submission in a secure cloud enclave, using remote access, VPN, or similar technology. All processing in the system would happen on a secure, remote server under control of the data owner, and the environment on the user’s end could be controlled via remote access.

---

protocols. Thus, users could potentially submit analyses over this connection and receive results in a secure manner. Decisions as to how to approach this environment will, of course, be up to individual agencies and the National Secure Data Service.

**How the Tool Works from a User Perspective**

We imagine researchers and analysts running analyses on the validation server will typically first run analyses on the Public Use File on their own personal or work computers. Once they have investigated the data and formed an analysis or analysis plan, the researcher or analyst would edit their code to fit the format our validation server requires and upload that code to our web interface. Currently, our interface supports SQL, but in the future, we plan to support multiple programming languages commonly used in the research community, such as Stata and R.

To submit their code, researchers first log in and are met with an Upload and Explore page, where they can enter their code. In the future, we hope to develop a landing page with an introduction to how the system works and links to learn more information about key concepts described in the previous section, such as differential privacy and privacy budgets. Researchers can view both their review and refinement budget and remaining balance and their public release budget and remaining balance. After uploading a file, the researcher can view the results of the analysis on the Public Use File with a prespecified amount of noise added by our privacy-preserving algorithms. The Results view allows the researcher to see the budget used, the total amount of error added, and the data for each command issued. Our hope is this information allows the researcher, who likely has not been exposed to a privacy budget previously, to learn about how noise addition and budgets work for their specific analysis before running the analysis on the confidential data.

Once the researcher is satisfied with the analysis commands submitted on the Public Use File, they can add the commands they are interested in running to the review and refinement queue and advance to the next page. Adding runs to the review and refinement queue will trigger that same analysis to run on the confidential data, though the researcher cannot see the results until they move to the next page. When implemented in a secure manner by a government agency, we imagine this next page to be the entry point into the secure environment; the results that could be viewed after this point would either be submitted remotely in a secure environment or by a person on site in a secure environment.
After clicking next, the researcher now sees the Review and Refine page, which again contains information about both their review and refinement budget and remaining balance and their public release budget and remaining balance. The researcher can also compare the results of their submitted analyses from the previous page (with privacy-preserving noise added) with the same results generated on the synthetic data. This allows the researcher to easily compare the difference between the two and further refine the analysis in the previous page if necessary. The result also provides a total error metric. In the Review and Refine tab, the researcher can also view a trade-off graph that shows the total error on the Y axis and the privacy budget on the X axis. The researcher can then determine the optimal trade-off of privacy versus usefulness for their specific task and choose the appropriate budget from a list of options. They can then click “add new version” to run a new analysis using the updated budget chosen and view the results. Each analysis run against the confidential data reduces the review and refine privacy budget, and the amount remaining is updated accordingly. In the future, we imagine researchers who exhaust their privacy budgets would have the opportunity to petition the data owner to add more budget on a case-by-case basis.

Once the researcher is done reviewing and refining their results, they can add the results they wish to publish to a final review and request queue and advance to the next page. On the following Request and Release page, researchers have the opportunity to review their final request queue, remove any analyses they do not wish to release, review the impact of the request on their overall remaining privacy budget, and submit a request to publicly release the results and spend the specified amount of the privacy budget. In the future, we imagine all submitted requests would be automatically cleared and e-mailed to the user for release. However, in the interim, we envision that a share of the requests would be manually reviewed by a statistical disclosure expert, and the share of manually reviewed requests might decrease as trust in the automated system grows.

A final fourth tab is available for researchers to view saved or past runs, in case they wish to build off past results or use past results to inform future analyses.

**Technical Infrastructure**

In this section, we describe the technical infrastructure at a high level and discuss how the community can set up, configure, and modify this system themselves.
How the Technical Infrastructure Works
The validation server technology system is composed of three parts: a front-end user interface for submitting analyses and viewing results, a back-end system for running analyses in a privacy-preserving manner, and an API to coordinate the communication between the front and back ends.

*Front End*
Researchers interact with the validation server system via a front-end user interface built on the React.js JavaScript library. The front-end interface facilitates user authentication, visual display, selection, refining, and release of data and privacy budgets via direct JavaScript calls to the API. The front-end interface also uses JavaScript to parse a command file and send that command to the API, though we may move that functionality to the API in a future release.

*Application Programming Interface*
User interactions between the front-end interface and the back-end differential privacy system are handled by an API built using the Django REST framework. The Django model objects reside in a MySQL database.

At the moment, new users must be added manually by an administrator to allow potential users to be screened. However, we hope to make this process more seamless in future iterations. User authentication is handled via Django REST’s built-in TokenAuthentication scheme. Once the user provides a username and password, the front end posts this information to the API token authorization (api-auth-token) endpoint, and the API sends the user’s token back to the front end if the authentication succeeded. This user token is submitted as part of the request header by the front end for all following requests to the API endpoints to verify the user’s permissions.

The API serves as the connection between the front end and the back-end Differential Privacy (DP) engine. Upon authentication, the user can upload a command file containing a transformation and table query. After parsing, the front end posts these queries to the command endpoint. The API then triggers the creation of a synthetic data run and starts the run on the back-end engine (an Amazon Web Services lambda function). The lambda function posts the results of this run to the synthetic-run-result endpoint, where the front end can request results for display on the web interface. Once the user decides which commands they want to add to their confidential run queue, the front end posts five runs to the confidential-data-run endpoint.
Creating these run objects, in turn, triggers five lambda functions that will return the confidential run results.

The front end requests the run results from the confidential-run-result endpoint and will display the run with the default epsilon value (1.0).

**Back End**

The back-end system receives user input for transformation and analysis queries from the API, and it coordinates running those queries against the synthetic and confidential Public Use File data tables. These data tables reside in a PostgreSQL database hosted on an Amazon Web Services (AWS) RDS server. The back-end system is deployed as an AWS lambda function, a “serverless” and event-driven compute platform. The lambda function is built on a Python version 3.8 runtime.

When the user uploads a new command file to the front-end interface and a synthetic data run record is created in the API database, the API triggers the back-end lambda function with an event payload that includes the transformation query (if supplied), analysis query, and the epsilon level at which to run the analysis query.

The lambda function first parses the event payload it received. If a transformation query was supplied, it is run against the synthetic data table with the `psycopg2` PostgreSQL driver for Python. The user-supplied analysis query is then run with the SmartNoise library of privacy-preserving algorithms at the given level of the epsilon parameter. The SmartNoise library returns a `pandas` data frame of the results of the query, along with a data frame containing the query’s level of accuracy. Both the result and accuracy data frames are parsed into JSON format and posted back to the synthetic data results endpoint of the API. The back-end system follows a similar process for running an analysis against the confidential data table.

**How Another Organization Might Set Up and Configure the System**

The validation server technology system is built and deployed in a cloud-native manner, hosted on the AWS cloud compute platform. The system is composed of four GitHub repositories. The repository for AWS cloud infrastructure ([https://github.com/UI-Research/validation-server-infrastructure](https://github.com/UI-Research/validation-server-infrastructure)) can be used as a guide to create a set of cloud resources for the overall validation server system. This repository includes an AWS CloudFormation template and a deploy script. This CloudFormation template creates an AWS Simple Storage Service that is
encrypted at rest, an AWS RDS PostgreSQL instance to hold the synthetic and confidential data tables, and an AWS Elastic Container Registry repository for storing the Docker images that power the back-end engine.

The codebase for the front-end interface is hosted in another repository (https://github.com/UI-Research/validation-server-frontend). This repository contains a Dockerfile for creating a containerized version of the application that can be run locally for testing or deployed on a server for public use. To manage the deployment of the front-end application, we use CircleCI, a continuous integration and continuous deployment (CI/CD) platform. When changes to the codebase are committed and pushed to this repository, the CI/CD system will rebuild and redeploy the application container on the server that hosts the staging site.

The codebase for the Django REST application programming interface (https://github.com/UI-Research/validation-server-api) similarly contains a Dockerfile for creating a containerized version of the API that can be run locally for testing or deployed on a server for public use. This repository also contains a `docker-compose` file that configures additional containers that make up the full API service. This includes a MySQL database container that holds the API data and a nginx container that is used as a proxy server for coordinating how users of the staging site access the front-end and API services. The `deploy.sh` script contains the steps for deploying this system using the `docker-compose` tool. This repository is also set up under a CI/CD platform, so when code changes are committed and pushed, the `docker-compose` stack is rebuilt and redeployed.

The codebase for the back-end engine (https://github.com/UI-Research/validation-server-engine) is built using AWS CodeStar and the AWS Serverless Application Model (SAM). This repository contains the source code for the AWS lambda function that handles running analysis queries in a privacy-preserving manner, along with a CloudFormation template for deploying the serverless application. When changes to the codebase are committed and pushed up to the repository, AWS CodeStar will package the SAM stack and deploy the serverless back-end engine.
How Another Organization Might Modify the System and Add Functionality

We purposefully separated the privacy-preserving validation server technology stack into separate containerized pieces so each could be modified with minimal disruption to the overall system.

The front-end interface interacts with the rest of the validation server system via the Django REST API. This means the interface could potentially be modified, and as long as what the API calls remains consistent (i.e., they retain their current general format), the rest of the system should continue to operate as intended. For example, the current interface only supports SQL statements, but we intend to eventually support additional syntax, such as Stata or R. This update can be done solely within the JavaScript front-end code without alterations to the API or back-end engine.

The privacy-preserving back end runs the queries received against a PostgreSQL database that contains the IRS Public Use File. This database could be swapped with any other confidential dataset with little impact on the rest of the system. As long as the queries received reference an existing table within the database, the SmartNoise engine should run as intended.

The back-end system currently only supports tabulation queries run via the SmartNoise library of privacy-preserving differentially private algorithms. This back end can be updated to handle additional algorithms as they are added to the SmartNoise library (e.g., for regression analyses). Or, an entirely different set of differentially private functions can be swapped into the back-end engine in place of using SmartNoise, if that is desired for a particular use. The `command` endpoint of the API layer was set to be a JSON field to remain flexible to different types of command input that may vary with the method or library used.

Limitations

The initial prototype system is currently built to the minimum standards for testing and improvement and therefore has several limitations. Currently, the tool only allows researchers to transform variables and produce various tables. It does not

- allow for more advanced analyses, such as regressions or other advanced statistical algorithms, or allow for the confidential data to have survey weights (weighted files must currently be adjusted to population values to work in the server);
- allow for researchers to join data (the data in the tool are the data that can be used);

- accept input other than SQL, though we know researchers are much more familiar with statistical programming languages like R and Stata;
- allow for any arbitrary transformation or table creation functions that researchers may be used to (only prespecified functions supported by PostgreSQL or the SmartNoise library are supported); or
- have robust learning libraries for privacy budgets, differential privacy, and other key concepts researchers are familiar with

In the future, we plan to address some key shortcomings that we heard in our user design sessions but were unable to incorporate in our initial prototype because of time and budget limitations. Some of these improvements, such as regressions and weighting, involve moving the field of differential privacy research forward. Others, such as supporting additional programming languages and adding learning materials, simply require additional time and budget. We also plan to share the tool with trusted audiences and our initial users to get additional feedback now that the prototype is fully functional, and we will add any additional items to our workplan for future development.