QueryShield: Cryptographically Secure Analytics in the Cloud

Ethan Seow
Boston University

Yan Tong
UC Santa Cruz

Eli Baum
Boston University

Sam Buxbaum
Boston University

Muhammad Faisal
Boston University

John Liagouris
Boston University

Vasiliki Kalavri
Boston University

Mayank Varia
Boston University

ABSTRACT

We present a demonstration of QueryShield, a service for streamlined, cryptographically secure data analytics in the cloud. With QueryShield, *data analysts* can advertise analysis descriptions to *data owners*, who may agree to participate in a computation for profit or for the greater good, provided that their data remain private. QueryShield supports relational and time series analytics with provable data privacy guarantees using secure multi-party computation (MPC). At the same time, it makes MPC accessible to non-expert users by offering a familiar web interface and fully-automated orchestration of cryptographic computations.

We devise three demonstration scenarios for conference attendees: (i) an interactive survey of private employment information to estimate the industry-academia wage gap in the data management community, (ii) a relational analysis that identifies credit score anomalies in sensitive customer data from multiple credit agencies, and (iii) a medical use case that assesses the effectiveness of insulin dose frequency in a patient cohort.

CCS CONCEPTS

• **Security and privacy → Cryptography**; • **Information systems → Data management systems**.

KEYWORDS

secure analytics; data privacy; multi-party computation

ACM Reference Format:
Ethan Seow, Yan Tong, Eli Baum, Sam Buxbaum, Muhammad Faisal, John Liagouris, Vasiliki Kalavri, and Mayank Varia. 2024. QueryShield: Cryptographically Secure Analytics in the Cloud. In *Companion of the 2024 International Conference on Management of Data (SIGMOD-Companion '24), June 9–15, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3626246.3654749

1 INTRODUCTION

We introduce QueryShield, a multi-party computation (MPC) service for secure outsourced analytics in the cloud. MPC [5, 11] is a cryptographic technique which allows multiple distrusting parties to collaboratively compute functions of their private data without revealing any information other than the output of the computation. The need for secure collaborative analytics emerges in various scenarios, where the societal or monetary benefit is magnified if multiple – often competing – entities allow analyses on the union of their sensitive data. As an example, consider a social scientist who wishes to study the racial wage gap in their community. Unfortunately, employers would be hesitant to reveal sensitive salary data to an untrusted party, as they may lose their competitive advantage. Other examples include medical institutions who may want to test the efficacy of a new drug by analyzing health records across their patient populations, or a group of banks that wish to perform fraud detection in financial data of common customers.

Recently, systems like Secrecy [3], Senate [13], Conclave [6], and TVA [4] have demonstrated that MPC can provide practical performance for relational and time series analytics on large inputs. Yet, deploying MPC computations currently requires cryptographic technical expertise and significant manual effort. QueryShield addresses this challenge by offering secure relational and time series analytics *as-a-service* on Secrecy and TVA. It features an easy-to-use frontend that allows data analysts to publish analyses they want to perform and data owners to join an analysis, after inspecting its definition and security guarantees. Once data owners have agreed to participate, the QueryShield backend automatically manages the configuration, deployment, and orchestration of MPC computations specified in the analysis.

We caution the reader that QueryShield is not a production-ready system. Rather, it is a proof-of-concept of a future where MPC analytics are accessible to non-expert users as a managed service that leverages the emerging multi-cloud and hybrid cloud environments. Our intention is to demonstrate the potentials of this technology and incentivize cloud providers to create the necessary services that would make systems like QueryShield widely available. To this end, we have devised three demonstration scenarios that include a secure wage gap survey, a multi-agency credit score study, and a mobile health analytics use case. We have made our code available on Github [10] and our short video on YouTube.[1]

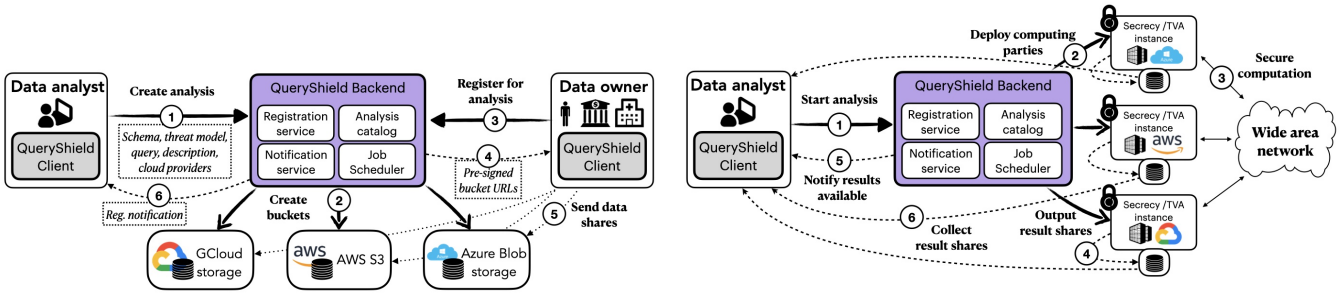
2 QUERYSHIELD OVERVIEW

QueryShield is a cloud service that enables users to access the Secrecy and TVA systems through a visual interface. Figure 1 provides an overview of the workflow. QueryShield supports two types of users: *data owners* and *analysts*. Analysts can create analysis tasks and publish their descriptions in a catalog maintained by the service.

SIGMOD-Companion '24, June 9–15, 2024, Santiago, AA, Chile

ACM ISBN 979-8-4007-0422-2/24/06
https://doi.org/10.1145/3626246.3654749

[1] https://www.youtube.com/watch?v=qcIhTWL8EKM



(a) Phase 1: Analysis creation and secure data sharing

(b) Phase 2: Query execution under MPC

Figure 1: QueryShield workflow overview: Phase 1 (left) starts with a data analyst creating and publishing an analysis in the analysis catalog ①, followed by QueryShield setting up buckets for storing secret shares ②. Data owners then browse the catalog, register for an analysis ③, receive pre-signed bucket URLs ④, and upload secret shares of their data to buckets ⑤. In Phase 2 (right), the data analyst submits an analysis for processing ①, QueryShield creates cloud virtual machines, and deploys the Secrecy/TVA software ② to start the secure computation ③. On completion, parties store the result shares in the output buckets ④, the analyst is notified ⑤, and retrieves shares of the result ⑥.

Data owners can view the catalog to decide whether they want to register for an analysis and contribute *secret shares* [11] of their private data. Analysts can then schedule analysis jobs for execution in available cloud providers, monitor the computation progress, and view the results when the job is done. Below we describe each step of the workflow in more detail.

2.1 Creating an analysis

To create a new analysis, the data analyst must provide some metadata (e.g., name, description, etc.), the schema of the private data needed for the analysis, the actual computation task, and the threat model. QueryShield supports tabular and time series data with attributes of four types: `INTEGER`, `VARCHAR`, `STRING`, and `CATEGORY` (for predefined lists of values). Analysts can express computations in SQL (for relational analytics) or using TVA's dataflow API [4] (for time series analytics). They also have the option to select cloud providers and threat models. QueryShield currently supports four cloud providers (AWS, Microsoft Azure, Google Cloud, Chameleon Open Cloud) and two threat models: one for semi-honest and one for malicious security (§2.4).

When an analysis is created, the next step is to request cloud storage for the secret-shared input and output data. The supported object stores are S3 (for AWS), Azure Blob Storage, Google Cloud Storage, and Ceph (for Chameleon). QueryShield requests one input bucket (for the data owners' shares) and one output bucket (for the result shares) per cloud provider in the analysis specification, both with *write-once* permissions. This configuration ensures that buckets are append-only and protects against malicious attempts to tamper the uploaded data (e.g., by external adversaries or even QueryShield users). Each new analysis gets its own dedicated buckets (initially empty) and input buckets are automatically discarded when the analysis is done. After all buckets have been created, QueryShield adds the analysis specification to the catalog and makes it available to data owners for registration.

2.2 Registering private data

Data owners can browse the analysis catalog to find information about analyses, including the requested data, threat model, and the actual task (e.g., a SQL query). When a data owner registers for an analysis, QueryShield generates one pre-signed URL for each input bucket created in the previous step. These pre-signed URLs are unique to the data owner and have an expiration time. All URLs provide *write-only* access to input buckets to guarantee that no one with the URL can read bucket contents (not even the data owner). The only parties that have read access to input buckets are the computing machines that run in the same cloud where the bucket lives (and this happens only after the analysis begins). Using pre-signed URLs, QueryShield simplifies the workflow by enabling data owners to safely upload secret shares of their data without the need to create accounts in the respective clouds.

Data owners can provide secret shares of their private data using CSV files or an online spreadsheet. In both cases, the input data schema must match the schema in the analysis specification, otherwise the frontend will throw an error. The QueryShield client generates secret shares of all input data locally, using *boolean* or *arithmetic* sharing (the choice depends on the computation). For example, to create a 3-party arithmetic sharing of the number 42, the client can generate three random shares `{-52, 39, 55}` that together add up to 42, but individually reveal no information about the original number. Similarly, to secret-share the ASCII character `'@'=0x40`, the client can generate the random shares `{0x71, 0x49, 0x78}` whose bitwise XOR equals to `0x40` [2]. Secret shares are distributed among cloud providers so that no single provider can reconstruct the original data. We emphasize that the QueryShield backend does not store any data; all shares are uploaded directly from the data owner's client to the respective bucket over HTTPS.

When uploading completes, QueryShield triggers an additional check to verify that each bucket contains all necessary secret shares from each data owner and, if not, it throws an error. This is to ensure the integrity of the input data before starting the execution.

2.3 Job scheduling and execution

Analysts can view the number of registered users and start the analysis on demand. To do so, QueryShield initiates a sequence of operations: it instructs cloud providers to provision computing resources (e.g., VMs), deploys the Secrecy/TVA software, performs

[2] In practice, QueryShield uses a ℓ-bit data representation (e.g., $\ell = 64$) and individual secret shares are uniformly distributed over all possible ℓ-bit strings.

query optimization [3], and starts execution. Computing parties perform the entire analysis under MPC, i.e., they operate directly on their shares and exchange messages with each other following a cryptographic protocol to end up with shares of the result. During execution, the analyst can monitor the job status and also gets notified when the job is done. Status updates on job submission, completion, or failure are also sent to registered data owners.

When execution finishes, each computing party adds its result shares to the output bucket. To reconstruct the true output of the computation, the analyst must pull the shares from the output buckets and combine them locally in the QueryShield client. There is also an option to keep results encrypted in the cloud.

2.4 Threat model

QueryShield's goal is to enable end-to-end secure computations on private data using untrusted infrastructure. To this end, the service acts as an orchestrator of the Secrecy and TVA systems and inherits their security guarantees, that is, it protects against computing parties and external (possibly malicious) adversaries in the honest-majority setting [3, 4]. Protecting against inference attacks from analysts (e.g., using Differential Privacy) is an orthogonal problem that is left as future work.

To the best of our knowledge, no cloud provider currently offers support for MPC computations. As a result, we have inevitably developed QueryShield as a centralized service. We acknowledge that this design choice is at odds with the decentralized trust model of MPC. To protect data from QueryShield itself, the backend functionality should be distributed among the participating cloud providers through separate managed services. In that case, each cloud service would be responsible for creating its local buckets and QueryShield would only be in charge of maintaining the analysis catalog. No other changes are required in the service architecture.

3 IMPLEMENTATION

Full details about our implementation are available on our project GitHub repository [10]. More details about the underlying MPC systems can be found in our prior work [3, 4].

Frontend. We have built the QueryShield frontend using React with Bootstrap. Secret sharing functionality is implemented with Axios and the JavaScript Web Crypto API. Both secret sharing and output reconstruction are done locally by the frontend via direct communication with the cloud providers. This way, plaintext input data never leave the data owner's machine and output data are only revealed to the designated analyst.

Backend. The QueryShield backend consists of two discrete components. The first one is a Google Firebase database that stores analysis metadata, e.g., the schema, name, and query provided by a data analyst. The second component is a scheduler that hosts the QueryShield API and coordinates a series of tasks (such as VM creation, job execution, and status updates) throughout the course of an analysis. The QueryShield API is implemented as a Python Flask RESTful service. It provides several key endpoints:

`/create_job`: Upon receiving a request, the QueryShield backend uses each cloud provider's SDK (AWS's Boto3, GCP's Client Library, and Azure's Python SDK) to create buckets for the analysis job.

`/register_data_owner`: The backend obtains pre-signed URLs for each bucket and provides them to the data owner. Each data owner receives their own set of URLs, preventing malicious activity or re-use by other data owners.

`/submit_job`: The backend generates an Ansible playbook for the specified query, which we employ to automate the creation of EC2, Google Compute Engine, and Azure VM instances. Once created, these instances access their respective secret shares from the cloud buckets and execute the query under MPC.

`/get_status`: The service checks the current status of the analysis and provides detailed updates. Possible statuses include `Creating VMs`, `Deploying TVA Core`, `Running Experiment`, etc. Once complete, this endpoint will return either `Success` or `Fail`. Data analysts can view these status updates on the QueryShield frontend.

4 DEMONSTRATION SCENARIOS

Attendees will have the opportunity to interact with QueryShield and choose to act either as data analysts or data owners. First, we will provide the attendees with an overview of MPC and we will showcase QueryShield's functionality. After log in, the attendees will be able to participate in one of the predefined demonstration scenarios we describe next or create their own data analysis tasks. The predefined scenarios include (i) a secure analysis of the wage gap between attendees working in industry and those working in academia, (ii) a relational credit score analysis query, and (iii) a time series analysis use case on mobile health data.

4.1 Live Survey of Industry-Academia Wage Gap

Our first scenario is inspired by a real-world use case of MPC that concerns quantifying the gender and racial wage gap in the Boston area [1]. The 2021 analysis was performed on private data contributed by 156,000 employees and its results are publicly available in a report by the Boston Women's Workforce Council. In our scenario, we will solicit private information from conference attendees to compute the industry-academia wage gap in the data management community, as represented by SIGMOD participants. In particular, our survey will ask attendees to act as data owners and contribute their salary, highest academic degree earned, years since graduation, work location, and whether they are employed in industry or academia. Since the answers to some of these questions may be sensitive, the frontend will locally create secret shares of responses and distribute them among computing parties. As a result, the sensitive data will never leave the participant's device in the clear. Attendees will further be able to inspect the contents of the generated cloud buckets and verify that their data are converted to random shares. Once enough attendees have agreed to participate, we will run a statistical analysis to calculate the wage gap between academics and practitioners, grouped by years of experience, education levels, and location. We will make the results of the analysis available online on the last day of the conference.

4.2 Relational Analytics: Credit Score Anomaly

Our second scenario will guide attendees to perform relational analytics with QueryShield and compute a SQL query against a secret-shared database. In particular, we will use the *Credit Score*

query that has been included in various relational MPC works [3, 13]. The analysis finds people who have a large discrepancy in credit score across different reporting agencies in a given year. We provide this example to demonstrate a query that – if not for MPC – would be extremely difficult to practically implement. The only information the query returns is a list of user IDs meeting the selection criteria. Neither their actual credit score from any particular agency nor any intermediate values (MIN, MAX) are revealed. The query is shown below.

```
SELECT S.ID FROM (
    SELECT ID, MIN(CS) as cs1, MAX(CS) as cs2
    FROM R
    WHERE R.year=TARGET_YEAR
    GROUP-BY ID ) as S
WHERE S.cs2 - S.cs1 > THRESHOLD
```

In this scenario, attendees will first act as the data analyst who wants to perform the credit score analysis. They will create the data schema, provide the SQL query, select the threat model, indicate the Cloud providers they want to use, and publish the analysis in the catalog. Next, they will switch role and contribute data for one or more of the credit agencies. As a data owner, they will be able to browse the available analyses in the catalog, register, and share data by using QueryShield's CSV upload feature or by inserting data records manually. Once the query is submitted, attendees will be given access to the Cloud providers' web consoles to get a glance of what happens under the covers of QueryShield. If desired, they will also be given the opportunity to act as an adversary and try to gain access to secret data. For example, they may choose to inspect network traffic or log into one of the computing party instances. Finally, they will see the results and verify the correctness of the output by running the query in the clear.

4.3 Time series Analytics: Glucose Monitoring

In our third predefined scenario, attendees will act as medical investigators who wish to perform mobile health analytics on private time series data. Specifically, they will be asked to use TVA's [4] dataflow API to express a glucose monitoring analysis task. We will provide a template query that uses a session window to identify a patient's *eating periods* as consecutive time intervals during which the patient's reported glucose measurement exceeds a threshold. The query then applies a window aggregation to compute the total number of insulin doses during each eating period. This template query can be expressed in the TVA dataflow API as follows:

```
// Define the time series data schema
TS ts = get_shares({"[PATIENT_ID]", "TIMESTAMP",
                    "[GLUCOSE]", "INSULIN", "TOTAL"});

// Mark eating periods per patient where [GLUCOSE]
// exceeds `5` and aggregate total insulin doses
TS doses = ts.keyBy("[PATIENT_ID]")
             .threshold_window("[GLUCOSE]" , 5)
             .aggregate("INSULIN", "TOTAL", Agg::SUM);
```

Note that column names in brackets (e.g., `[GLUCOSE]`) indicate boolean secret sharing, whereas unbracketed column names are arithmetically shared. Attendees will be asked to change the query definition to try out other time series operators supported by TVA, such as *tumbling windows* that group data within fixed, time intervals (for example, to calculate a daily average of some quantity), and *gap session windows* that identify periods of activity followed by periods of inactivity (for example, pedometer data with a gap of five minutes might distinguish walking from sitting). As in the previous scenario, attendees will be able to verify the correctness of their analysis results and act as adversaries, if they wish.

5 RELATED WORK

Other systems that support relational queries under MPC include Conclave [6] and Senate [13]. Waldo [2] supports secure time series analytics using function secret sharing (FSS). An alternative to MPC is to use hardware enclaves, as in CCF [9] and SCONE [7]. Enclaves are faster than MPC solutions but susceptible to side-channel attacks. Many cloud providers now offer "clean rooms" for collaborative analytics [8, 12, 14] that do not rely on enclaves; however, these services typically require trusting the provider.

6 CONCLUSION

In this paper, we have presented QueryShield, an accessible service for cryptographically secure multi-party computation. QueryShield allows data owners to contribute private data to joint analyses in the cloud while protecting the data from untrusted or unauthorized entities. It supports relational queries in SQL and time series analytics with a familiar dataflow API.

ACKNOWLEDGMENTS

This work was partially supported by the NSF SaTC Core Medium Award #2209194 and a gift by Robert Bosch GmbH.

REFERENCES

[1] Boston Women's Workforce Council. 2021. Addressing the Gender and Racial Wage Gap. https://www.bu.edu/articles/2021/using-data-science-to-address-the-gender-and-racial-wage-gap/.
[2] Emma Dauterman *et al.* 2022. Waldo: A Private Time-Series Database from Function Secret Sharing. In *IEEE S&P*. 2450–2468.
[3] John Liagouris *et al.* 2023. SECRECY: Secure Collaborative Analytics in Untrusted Clouds. In *USENIX NSDI*. 1031–1056.
[4] Muhammad Faisal *et al.* 2023. TVA: A Multi-party Computation System for Secure and Expressive Time Series Analytics. In *USENIX Security*. 5395–5412.
[5] Nigel Smart *et al.* 2024. Multiparty Computation: To Secure Privacy, Do the Math: A discussion with Nigel Smart, Joshua W. Baron, Sanjay Saravanan, Jordan Brandt, and Atefeh Mashatan. *Queue* 21, 6 (2024), 78–100.
[6] Nikolaj Volgushev *et al.* 2019. Conclave: Secure Multi-party Computation on Big Data. In *ACM EuroSys*. 1–18.
[7] Sergei Arnautov *et al.* 2016. SCONE: Secure Linux Containers with Intel SGX. In *USENIX OSDI*. 689–703.
[8] Google. 2024. BigQuery Data Clean Rooms. https://cloud.google.com/use-cases/data-clean-rooms. Accessed: January 2024.
[9] Heidi Howard *et al.* 2023. Confidential Consortium Framework: Secure Multiparty Applications with Confidentiality, Integrity, and High Availability. *Proc. VLDB Endow.* 17, 2 (2023), 225–240.
[10] BU CASP Systems Lab. 2024. QueryShield GitHub. https://github.com/CASP-Systems-BU/queryshield-demo.
[11] Yehuda Lindell. 2020. Secure Multiparty Computation. *Commun. ACM* 64, 1 (2020), 86–96.
[12] Microsoft. 2024. Confidential Data Clean Rooms. https://techcommunity.microsoft.com/t5/azure-confidential-computing/confidential-data-clean-rooms-the-evolution-of-sensitive-data/ba-p/3273844. Accessed: January 2024.
[13] Rishabh Poddar *et al.* 2021. Senate: A Maliciously-Secure MPC Platform for Collaborative Analytics. In *USENIX Security*. 2129–2146.
[14] Amazon Web Services. 2024. AWS Clean Rooms. https://aws.amazon.com/clean-rooms/. Accessed: January 2024.