RosemanLabs

PRIVACY BY DESIGN

Technical Sheet

Virtual Data Lake

Work together on sensitive data,
without revealing your input

Executive Summary

This document describes our main product, the Virtual Data Lake (VDL) and the technology underlying it. The VDL is a flexible data collaboration platform with which multiple parties can collaborate on data (join, query, analyse data sets) without revealing input data to each other. Data scientists can interact with the VDL via Python scripts, as if it were a normal database.

The VDL is based on MPC (Multi-Party Computation) technology. We use Shamir secret sharing based MPC, providing strong confidentiality and correctness guarantees. The VDL takes full advantage of modern server hardware by exploiting parallelism while avoiding thread contention. This leads to a runtime performance that is close to linear in the number of processor cores.

Throughout all steps of the computation, the confidentiality of the data is guaranteed; neither analysts interacting with the VDL, nor system administrators managing one of the VDL servers, have access to the input data or any of the intermediate results. The outcome of the analyses can be disclosed to one or more designated parties.

Although we opted to use logistic regression to provide quantification, in principle any machine learning model can be used in the VDL; however, some models, such as very large or deep neural networks, will result in excessive runtime performance.

1 MPC technology

Secure multi-party computation (MPC) is a privacy technology that lets multiple parties collaborate by enabling computations on their combined data, in such a way that the parties' inputs remain mutually secret; only the result of the computation becomes known (to one or multiple designated parties).

MPC has a strong theoretical foundation based on four decades of academic research. As such, it immediately sets itself apart from “hardware engineering”-approaches to secure computation, such as trusted execution environments (TEEs).

At some point in the past decade, a turning point was reached for MPC, in that the runtime performance of MPC (which was a rightful concern in the past) improved to a point where it no longer forms a barrier for most practical use cases. This turning point has been reached through technology advances in computing and networking hardware, as well as through several theoretical breakthroughs. Several members of Roseman Labs’ team of cryptographers have played a role in some of these breakthroughs.

MPC belongs to a broader class of privacy technologies originating from modern cryptography; other technologies within this class are, for example, fully homomorphic encryption (FHE) and zero-knowledge proofs (ZKPs). Roseman Labs’ main focus is on MPC, because we are convinced that MPC is currently the privacy technology with the best overall "scorecard", when taking security, runtime performance, and versatility into account.

Roseman Labs currently applies Shamir secret sharing based, passively secure MPC, typically using three computational nodes, which can be deployed on-premise or in the Cloud (or as a mixture of the two). Although the number of computational nodes is deliberately kept low for cost and runtime performance reasons, the number of clients that can connect to the MPC servers (to securely provide input or consume output) is not limited by this figure. By leveraging modern container technologies like Docker and Kubernetes, we provide a smooth deployment process of our solutions in our clients’ environments.

Roseman Labs’ main product, the Virtual Data Lake, is powered by Cranmera, our production-grade MPC engine. Cranmera has been built from the ground up to take full advantage of the power of modern server hardware. By exploiting parallelism while avoiding thread contention, we achieve speedups linear in the number of processor cores, enabling Cranmera to deliver the best-in-class MPC runtime performance.

2 The Virtual Data Lake

The Virtual Data Lake (VDL) enables organisations to break down data silos, by offering the possibility to extract insights from virtually combined data sets without the need to reveal their sensitive input data to each other. The VDL has several benefits over traditional data collaboration:

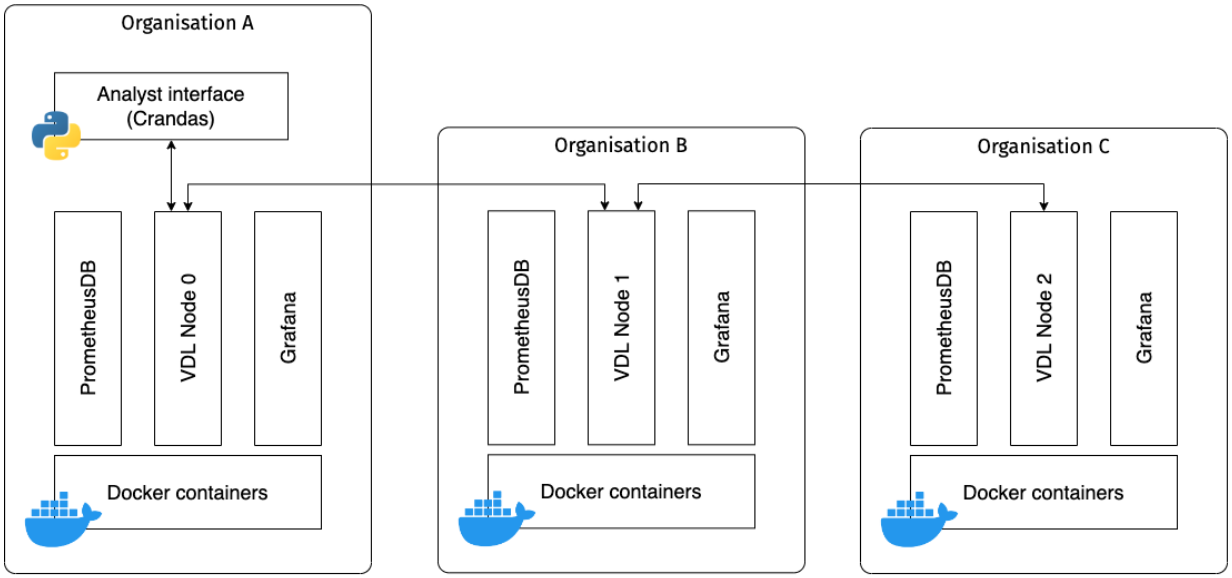
- Confidentiality: Input data is never revealed – Sensitive data is encrypted locally at each organization. As part of the encryption process, the data is partitioned into parts that do not disclose any information. These parts are then sent to the different nodes in the virtual data lake.
- No data sharing: Data is only combined virtually – Due to the nature of MPC (Multi-Party Computation) the data is only combined virtually as part of the execution of a query. Only the outcome of this query is presented to the analysts, and no input data is disclosed.
- Python interface: Data analysts can interact with the VDL as if it was a normal database – The VDL is an abstraction layer on top of our Cranmera engine. This enables data scientists to interact with the VDL like they would with any other database, giving them access to state of the art cryptography without the need for any additional training or resources. This interaction is enabled by our Python library Crandas (explained in more detail in section 2.2).
- Strong purpose binding: Types of queries, output and viewers are pre-agreed and signed off – A collaboration based on the VDL is governed by a group of data owners that agree upon the queries that can be executed, the output that can be generated, and the individuals that can access the VDL. This ensures that no data can be extracted beyond the scope of the collaboration. In terms of the GDPR, this provides strong purpose binding and control.

2.1 Architecture & deployment scenarios

As outlined in Section 1, a deployment of the Virtual Data Lake consists of at least three nodes. These nodes are based on Docker technology and can be deployed both on-premise and in cloud environments. Due to the nature of MPC, segregation of duties (SoD) needs to be in place between the administrators of those nodes. This can be achieved in three different deployment scenarios:

- Centralized – All servers deployed in a single organization; segregation of duties implemented between administrators within the same organization.
- Distributed – All servers deployed in logically and legally separated environments. Thereby implicitly implementing SoD as the administrators are part of different organizations. A distributed deployment can exist of a combination of cloud and on-premise hosted servers.
- SaaS – All servers deployed in a SaaS environment that is managed by Roseman Labs; segregation of duties implemented between Roseman Labs administrators (and cloud providers).

A schematic representation of a VDL deployment is outlined in the image and description below.



- VDL Node — A docker container containing the VDL server application. This container requires connectivity with the other VDL nodes and the Crandas library used by the analysts.
- Analyst interface (Crandas) — Explained in more detail in 3.2
- PrometheusDB — Used to collect metrics from the VDL Node and store them in a database.
- Grafana – Used to create monitoring dashboards based on the data collected in the PrometheusDB. For example, monitoring resource usage by the VDL node.

2.2 Analyst interface – Crandas

Analysts interact with the VDL through a Python library called Crandas. This library mimics the widely used data science library Pandas and provides a familiar interface to the user.

Crandas serves as an abstraction layer between the multi-party computation performed by the VDL nodes and the analyst. Analysts can develop scripts in Python as if they are working with Pandas (or other data science libraries) on locally available data sources. Crandas translates these Python operations into commands executed by the VDL nodes. The commands are sent to the node that is managed by the analyst's organization, this will orchestrate the computation by the different VDL nodes and return the result to the analyst.

2.3 Features of the VDL

Features of the VDL are continuously extended. For an up-to-date overview, visit our online documentation portal: rosemanlabs.com/rldocs. Current functionalities include a wide range of operations for statistics and machine learning purposes.

2.4 Generic runtime performance description

The table below lists the generic performance metrics for the main functionalities which are available within the VDL. These metrics were obtained in a deployment on a Kubernetes (K8s) cluster hosted by DigitalOcean. This cluster consists of three nodes which have 16 CPUs and 128GB of RAM each (m3-16vcpu-128gb). The resulting key runtime performance metrics are:

Logistic regression - training model using 10 iterations

Rows	Timing (16 cores /node)
100	2.2s
1000	9s
10K	79s
100K	783s / 13min

Logistic regression - Inference probabilistic

Rows	Timing (16 cores /node)
100	0.2
1000	2s
10K	20s
100k	210s

Logistic regression - Inference binary

Rows	Timing (16 cores /node)
100	0.2s
1000	2s
10K	19s
100K	199s

Join (two tables containing the number of rows listed)

Rows	Timing (16 cores /node)
100	0.2s
1000	0.2s
10K	0.7s
100K	6.8s
1M	69s

Filtering (a single table containing the number of rows listed)

Rows	Timing (16 cores /node)
100	0.02s
1000	0.03
10K	0.1s
100K	0.7s
1M	6.2s

3 Threat model, confidentiality and integrity guarantees

The VDL servers use arithmetic secret sharing based on the BGW (Ben-Or, Goldwasser, Widgerson) protocol with **Shamir secret sharing**. In a set-up with three servers, this is secure against at most one passively corrupted party (i.e. an honest-majority assumption). This means that no single server knows the data, but a majority of servers could in principle learn the data by combining their information. To avoid this scenario in practice, we assume that the servers (and their independent administrators) do not collude. This may in practice be realized by distributing the server admin rights over three different collaborating parties.

In more detail, every value *v* in the Virtual Data Lake is stored in a distributed way by storing so-called **secret-shares** *v*_1, *v*_2, *v*_3 at the respective servers. The values *v_i* have the property that any *v_i* does not reveal any information about *v* (given a single *v_i*, all values for *v* are equally likely), but *v* can be uniquely computed from *v*_1 and *v*_2, from *v*_1 and *v*_3, or from *v*_2 and *v*_3.

In the Virtual Data Lake, like in a relational database, data is organized in tables, on which various operations can be performed. Computations (e.g., filtering, joining) take one or more private input tables and perform a computation that results in a private output table. All computations are carried out using 1-out-of-3 passively **secure multi-party computation (MPC)**. This means that no single party learns any information about the inputs, outputs, and intermediate values of the computation, apart from meta information that is explicitly deemed public, such as which operation is performed, the name of the table, the number and type of columns, and the number of rows.

Furthermore, the MPC operations are implemented in such a way that the time it takes to complete that operation only depends on public information. This means that an attacker cannot learn information on secret data based on timing measurements.

3.1 Roles

A VDL deployment involves various roles, which we describe here for the sake of clarity.

We define the **participants** to be the organizations participating in the data collaboration. An **input party** is a participant that provides data as input to the computation; a **compute party** is a participant that runs and administers a VDL node (server), and an **output party** is a participant that is eligible to receive computational results. Note that participants could fulfil multiple roles simultaneously, e.g. a participant could be an input party as well as a compute party. A **client** is a natural person, affiliated to one of the participants, that runs **queries** to upload data, perform data analyses and/or download results, typically from a Python environment.

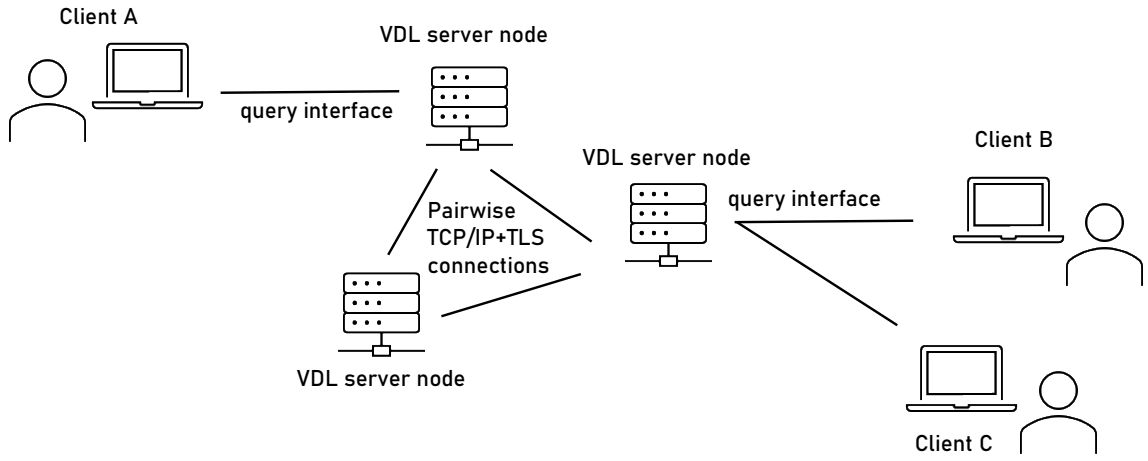
3.2 Client authentication and query authorization

The **query-authorization layer** ensures that a client can only perform queries that have been explicitly authorized by one or more **representatives** of the participants. The authorization mechanism is flexible in that it can authorize specific queries, or authorize entire classes of queries via a templating mechanism. The query-authorization layer serves as a powerful security control. For example, it will prevent uncontrolled data leakage in a 'rogue employee'-scenario, in which an employee of one organization tries to steal or browse through the entire data set from another organization.

A representative of the participants is a natural person that is given the authority to authorize queries and should only do so if running the particular query would be in the interest of the participants, and has a legal basis (e.g. under the GDPR). The representative can authorize a query by digitally signing it. Hence, the representative (or group of representatives) forms the link between the "human world" (involving interests, legal aspects, etc.) and the VDL, which will execute a query if and only if the signature is valid (but does not know anything about aspects of the "human world").

3.3 System and networking topology

In the figure below we show an exemplary VDL setup; the items shown in the figure are discussed in the text further below.



The core of the VDL system consists of three VDL server nodes. The servers are pairwise interconnected via secure channels (TCP/IP connections secured with Transport Layer Security v1.3).
A client connects via the query interface to the VDL, which uses JSON-over-HTTPS. Hence, the pairwise server connections as well as the query interface are secure connections protecting against malicious outsiders.

To connect as a client via the query interface, you only need to connect to a single VDL server node. This does *not* imply, however, that this VDL server node can inspect the data being communicated over this interface; on the contrary, all uploaded and downloaded data is hidden from the individual VDL servers using the VDL input/output module. This module masks uploaded and downloaded data such that the servers cannot learn which data was uploaded or downloaded. Data uploaded via the input/output module results in that data being available in Shamir shared form at the VDL server nodes, to which the above-mentioned security measures apply. Downloading data achieves the reverse: Shamir secret shared data stored at the VDL server nodes is converted (by means of an MPC protocol) to a masked representation that is unmasked only at the client's end.

If the client using the query interface is affiliated to a participant, that is also a compute party, the client can connect to the query interface via the participant's Intranet. On the other hand, by exposing the query interface to the Internet (with appropriate network security controls), other participants that are not themselves a compute party, can connect to the VDL via that server node. Recall that the compute party to which these other clients (affiliated to other participants) connect will not be able to inspect the data uploaded by these clients.

3.4 Data integrity: assumptions and guarantees

Theoretically, the BGW+Shamir MPC protocol provides correctness, meaning that given correct inputs, the computational result(s) will be correct. Note that the input parties could always deliberately provide incorrect inputs, which cannot be avoided a priori (a posteriori, a party who provided false inputs could be penalized by the other parties).

The correctness guarantees above are under the assumption that, at each VDL server node, the server executable (the application binary) is not tampered with. Likewise, we assume that the cache files that the servers store locally on disk are not tampered with. Both aspects can be guaranteed in practice using standard server-administration access control policies. Furthermore, we need to assume the correctness of the software implementation. Our confidence in the implementation-correctness of our software is based on repeated and automatic testing, and periodic external code audits (one of which is underway by the Dutch National Cyber Security Center of which results will be available in the coming months).

3.5 Conclusions

The VDL solution offers strong confidentiality guarantees, namely as long as the majority of the VDL server nodes is 'honest' (i.e., do not collude), no single participant, client nor server administrator can learn any information about the data that stored in and manipulated with the VDL, except for public meta-data, and except for the computational results that are explicitly revealed to one or multiple clients (assuming they are authorized to obtain these results).
Data-integrity is guaranteed under the assumption that the integrity of the input data is guaranteed, the software implementation is correct, and its deployment is not tampered with.

Ian Wachters

Ian.wachters@rosemanlabs.com

+31 6 20 43 62 47

RosemanLabs

PRIVACY BY DESIGN