Non-elastic restricted Secure SQL Engine

D3.4

Project reference no. 653884

February 2017

European Commission

Horizon 2020
European Union funding
for Research & Innovation

Document information

Scheduled delivery	01.03.2017
Actual delivery	01.03.2017
Version	1.0
Responsible Partner	Cybernetica

Dissemination level

Public

Revision history

Date	Editor	Status	Version	Changes
14.09.2016	B. Portela, F. Maia	Draft	0.1	ToC
20.12.2016	D. Bogdanov	Draft	0.2	Added content
26.01.2017	K. Tarbe, V. Sokk, F. Maia	Draft	0.3	First draft
09.02.2017	K. Tarbe, V. Sokk	Draft	0.4	Final draft
28.02.2017	K. Tarbe	Draft	1.0	Address the reviews

Contributors

Dan Bogdanov (CYBER)
Karl Tarbe (CYBER)
Ville Sokk (CYBER)
João Paulo (INESC TEC)
Francisco Maia (INESC TEC)
Bernardo Portela (INESC TEC)

Internal reviewers

F. Maia (INESC TEC)
Bruno Ferreira (Maxdata)
Paulo Sousa (Maxdata)

Acknowledgements

This project is partially funded by the European Commission Horizon 2020 work programme under grant agreement no. 653884.

More information

Additional information and public deliverables of SafeCloud can be found at http://www.safecloud-project.eu

Glossary of acronyms

Acronym	Definition
SQ3	Secure Queries layer Solution 3: Secure multi-cloud application server
SQL	Structured Query Language

Table of contents

1 Introduction

In the initial design, SafeCloud was proposing a single Secure SQL Engine component, which is the main focus of the current deliverable. However, as plainly visible in the previous deliverables D3.1, D3.2 and D4.1, we are now proposing three independent solutions for secure SQL processing. These solutions target different trust models and aim to offer a broader set of options for applications that want to run secure data processing in the cloud.

If we analyse the three solutions proposed, two of them share a similar high-level architecture while the third one is built independently following a radically different approach. In particular, solutions 1 and 2 are built on top of a secure NoSQL database and solution 3 is based on the Sharemind secure computing engine[1].

Considering the dependence of solutions 1 and 2 in the Secure Key Value Store, which is still in its initial release (described in deliverable D3.3), the integrated prototypes for these solutions cannot be presented yet. Instead, they will be presented in detail in deliverables D3.6 and D3.8. Nevertheless, we briefly address them under the SQL over Encrypted NoSQL designation by describing their overall architecture and how the integrate with the Secure NoSQL Store. This is the subject matter of Section 2.

Having said this, in this document we will focus on the preliminary prototype for solution 3. This prototype represents the initial release of a non-elastic restricted secure SQL engine based on multi-party computation techniques. The extent of the SQL support, the implemented features as well as how to set up the prototype demonstrator are described in Section 3. We conclude the deliverable with Section 4.

2 SQL over Encrypted NoSQL

2.1 Overview

The design of all SafeCloud secure processing solutions focus on clearly defining concrete trust models and adjust each solution to adequately match them. For solutions 1 and 2 this meant defining two different sites: one site that is considered to be trusted by the application, which means that data can be processed in clear in this site, and an untrusted one where data must be protected either for storage or processing.

In the present discussion, we will group these solutions as they both rely on a Secure Key Value Store. The main design decision is to have this store deployed on the untrusted site (a cloud service provider) and offer different compromises between its data protection and processing power according to different possible privacy preserving techniques that our solutions support. Stronger privacy levels require more processing power in the trusted site as data becomes opaquer to the untrusted site NoSQL store. Conversely, if we lower the desired privacy level we will be able to equip the NoSQL store with more powerful processing capabilities and, ultimately, achieve better system performance and scalability since we can take better advantage of third-party untrusted cloud infrastructures.

Following this design, we are able to separate data storage and processing in the cloud provider from SQL query processing and transaction management. These are done in the trusted site (typically where the client application is running) and leverage previous work as described previously both on deliverables D3.1, D3.2, and on the SafeCloud project proposal. In detail, previous research results led to the design of a query engine that is able to parse SQL statements and translate them into NoSQL (HBase-like)

[1] Sharemind - http://sharemind.cyber.ee/

operations. In SafeCloud, we have developed the Secure Key Value Store (extensively described in deliverables D3.3 and D3.5) to be compliant with this translator component. As a consequence, in the coming months we will be able to integrate both the query engine and the secure data store in order to provide SQL support also for solutions 1 and 2 of the platform.

2.2 Architecture and implemented features

The architecture for solutions 1 and 2 is depicted in Figure 1 and Figure 2 respectively. It is important to notice that it materializes the separation between NoSQL-like data processing on the untrusted environments and the SQL parsing and processing in trusted environments.

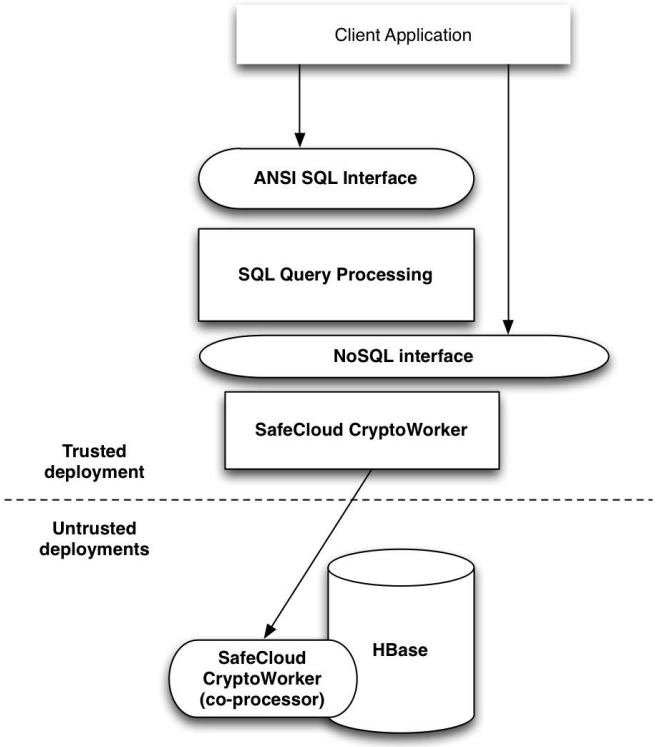

Figure 1: Solution 1 architecture

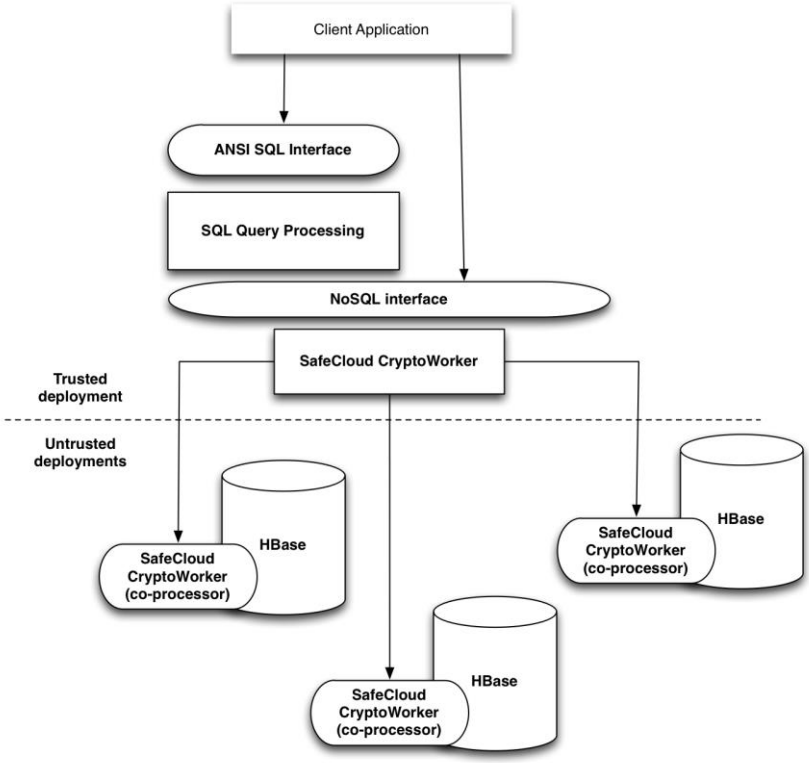

Figure 2: Solution 2 architecture

Presently, both solution 1 and 2 NoSQL Key Value Store components are functional and ready for integration. These are the subject matter of deliverable D3.3. In detail, we have a solution prototype instantiated with standard encryption CryptoWorkers, which is modular and allows changing the privacy preserving technique being used in a straightforward way. Moreover, a prototype for solution 2 NoSQL data store is running with multi-party computation protocols allowing to tolerate a compromised cloud provider. In order to attain the final version of the Secure Key Value Store prototype, the former requires the implementation of additional CrytpoWorkers (different privacy preserving techniques) while, for the case of the latter, full functionality is in place and the work in progress is mainly focused on performance.

2.3 Integration with use cases

The main use cases for the secure queries layer of the SafeCloud platform are those provided by Maxdata. These are SQL applications with specific privacy requirements. The main concern with respect to the integration of our solutions with the use cases is SQL compatibility and coverage. In particular, we need to guarantee that all SQL operations the use case applications may require are effectively supported by our platform. To this end we have run preliminary tests with initial versions of the use cases and the platform software components. These tests, that we are continuously perfecting, allow us to guarantee that the integration process will be manageable. Moreover, they allow us to timely detect any issues that may arise with such integration.

3 SQL over Secure Multi-party Computation

3.1 Overview

This section describes the implementation of privacy-preserving SQL query engine that executes all queries on cryptographically protected data using the Sharemind secure computing engine[2]. The SafeCloud project has successfully developed a module that translates SQL queries to be executed on the Sharemind computing engine, thus enabling queries on encrypted data with protection against the data host.
Furthermore, our solution also provides output privacy, meaning that no query will reveal an individual source record. Instead, the module provides various aggregations over the data. This is especially relevant in the SafeCloud medical use case pioneered by Maxdata, where confidential data is collected from several sources and no private record should ever appear in the output of the query.

3.2 Architecture

We have not deviated from the general architecture proposed in D3.1 and more detailed architecture described in D3.2 and D4.1. We have a client proxy which handles translating the SQL operations and communication with the Sharemind deployment. The architecture is shown on Figure 3.

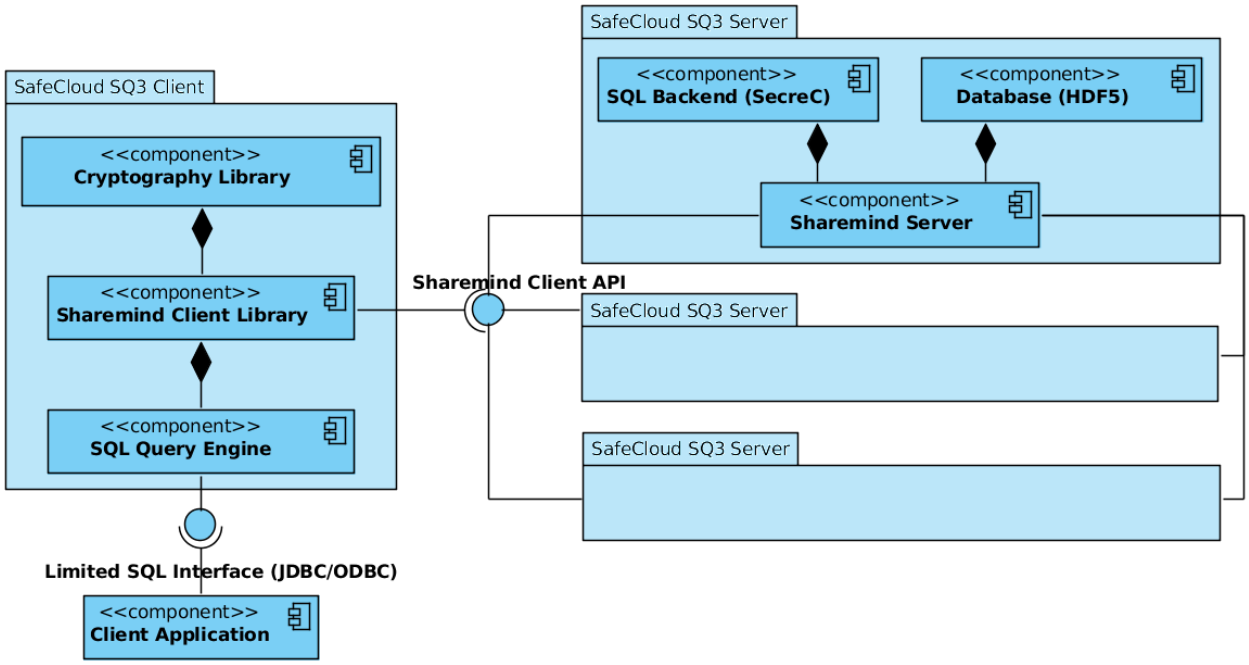

Figure 3: Solution 3 architecture

Internally it made sense to refine the front end of Solution 3 even more. One concern is handling SQL queries, another is to provide a machine interface for it, because the machine interface is involved with data types and some complexity that is unrelated to executing SQL queries. Therefore, we decided to develop two executables: one for testing the SQL Query Client directly and another that emulates PostgreSQL server to leverage existing drivers for PostgreSQL server and allow easy integration with existing applications. These alternatives are depicted on Figure 4.

[2] Sharemind - http://sharemind.cyber.ee/

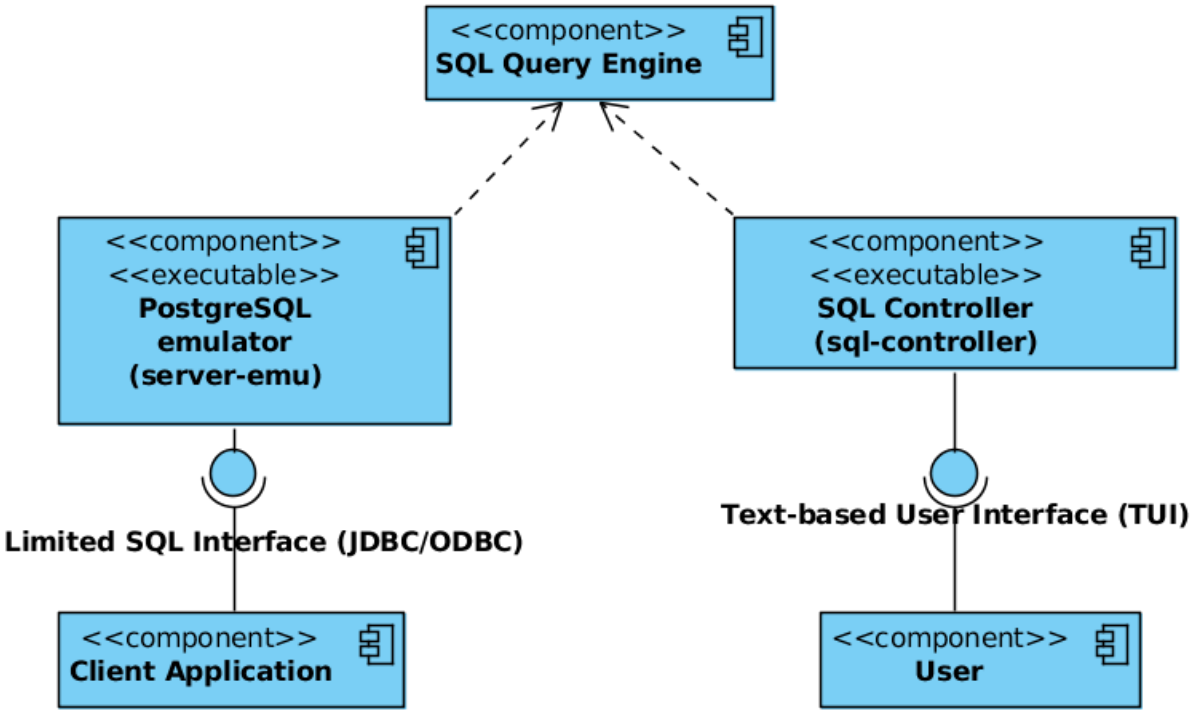

Figure 4: A closer view of Solution 3 frontend architecture deployment

The Client Querying Server (client proxy) must be in the trusted environment. The Sharemind deployment must be hosted by three non-colluding parties. One possible deployment can be seen on Figure 5.

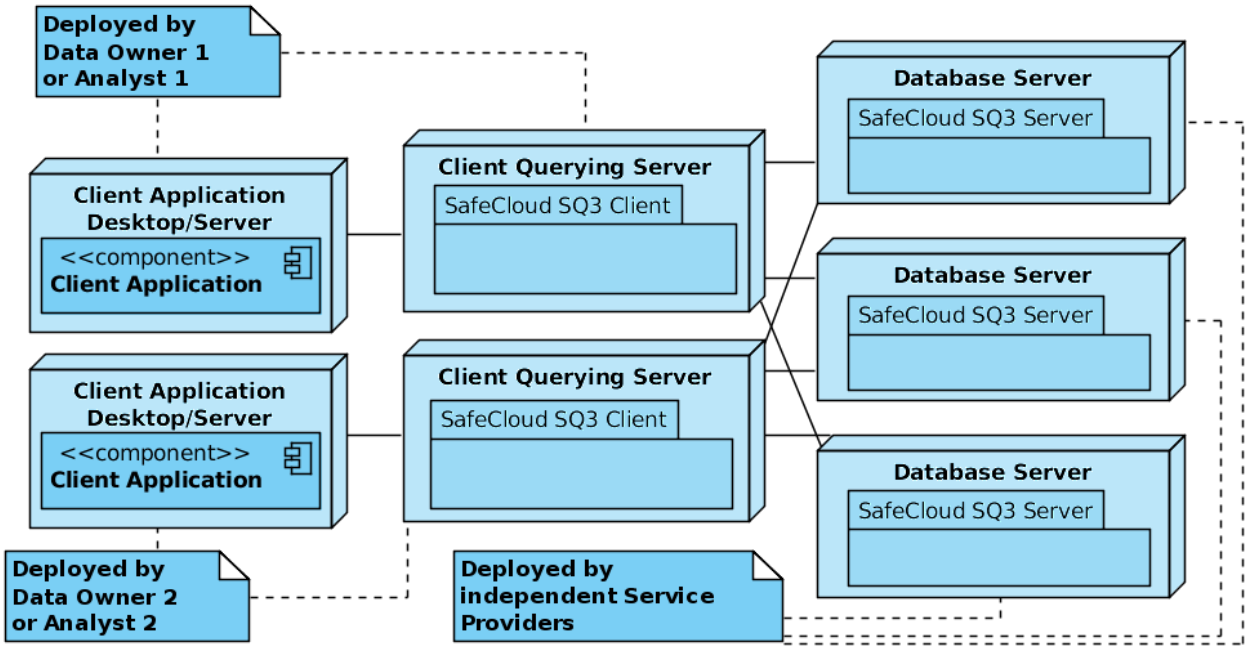

Figure 5: Typical deployment of Solution 3

3.3 Current state

The Sharemind framework and the Sharemind Analytics Engine (SecreC) existed before SafeCloud. SQ3 is a new frontend on top of Analytics Engine to support SQL. It is an

alternative user interface next to the Rmind[3] frontend. The SQ3 Server consists of the Sharemind Application Server with the Sharemind Analytics Engine installed.
SQ3 supports numeric data types and bounded length strings and most of the Data Manipulation Language. More detailed information about current status of SQL support is given in D3.5.

3.4 Setup and usage

Setting up SQ3 consists of a few larger steps.

1. Set up Sharemind.
2. Install Sharemind Analytics Engine backend into Sharemind cluster.
3. Set up the client application to use Client Querying Server for queries.

Right now, setting up SQ3 is a little more complicated than it could be, but in the future, there will be tools to assist with steps 1 and 2. Each of the three steps is explained in the following sections.

3.4.1 Setting up the Sharemind deployment

First you need to obtain a Sharemind licence and the binaries from https://sharemind.cyber.ee. Then find the independent hosts and deploy the binaries on them. Configure the Sharemind deployment. Detailed instructions for setting up the Sharemind deployment come with the Sharemind binaries. Make sure that modules *mod_tabledb* and *mod_algorithms* are enabled in the Sharemind configuration.

3.4.2 Deploying the Analytics Engine backend

First obtain the Sharemind Analytics Engine backend, which is a suite of SecreC scripts. Then compile the Analytics Engine backend with the SecreC compiler[4] and place the resulting *analytics-engine.sb* file into the scripts directory of Sharemind Application server (on all the computing parties).

3.4.3 Setting up the Client Query Server

Obtain the binaries. Make sure you have Sharemind controller configuration, which corresponds to your Sharemind deployment in *controller.cfg*. Client Query Server comes with two binaries: *server-emu* and *sql-controller*. The *server-emu* emulates PostgreSQL server and can be used with PostgreSQL drivers for various languages. The *sql-controller* executable is a simple user interface like *psql* is for PostgreSQL. Note that you can use *psql* with the *server-emu* if you like. The *controller.cfg* file, public keys of the servers and both keys of the controller must reside in the working directory when executing server-emu or sql-controller. Before using the Client Query Server for the first time, one needs to initialize the backend and that can be done by executing *"sql-controller --initbackend"*.

After the backend is initialized, one can use *sql-controller* directly or start *server-emu*. The *server-emu* binary will listen on port 5432 for PostgreSQL wire protocol and can be used with drivers for PostgreSQL server.

[3] Rmind - https://eprint.iacr.org/2014/512.pdf

[4] Comes with the Sharemind binaries, but can also be obtained from https://github.com/sharemind-sdk/secrec.

We have a number of SQL commands stored in text files used for demo purposes. They can be run with *sql-controller FILES...*, where *FILES...* is a list of files containing SQL commands (usually files ending with *.sql*).

4 Future work

For solutions 1 and 2 of the Secure Queries layer of the SafeCloud project, the integration with the SQL to NOSQL translating SQL Query Engine is in progress. For solution 3, the emulation of PostgreSQL server could use some improvements like coverage testing and validating with different drivers.