A framework for user centred privacy and security in the cloud

Architecture V1

<table>
<thead>
<tr>
<th><strong>Type (distribution level)</strong></th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Contractual date of Delivery</strong></td>
<td>31-10-2015</td>
</tr>
<tr>
<td><strong>Actual date of delivery</strong></td>
<td>27-10-2015</td>
</tr>
<tr>
<td><strong>Deliverable number</strong></td>
<td>D4.1</td>
</tr>
<tr>
<td><strong>Deliverable name</strong></td>
<td>Architecture V1</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>2.0</td>
</tr>
<tr>
<td><strong>Number of pages</strong></td>
<td>50</td>
</tr>
<tr>
<td><strong>WP/Task related to the deliverable</strong></td>
<td>WP4/Task 4.1</td>
</tr>
<tr>
<td><strong>WP/Task responsible</strong></td>
<td>OFFIS/EURECOM</td>
</tr>
<tr>
<td><strong>Author(s) (in alphabetical order)</strong></td>
<td>Frédéric Brouillé, Thierry Chevallier, Monique Damiot, Véronique Font (AKKA), Kaoutar Elkhiyaoui, Melek Önen (EURECOM), Wissam Mallouli, Antonio M. Ortiz (MTI), Stefan Janacek, Sven Rosinger (OFFIS), Cédric Brandily, Grégory Cunha, Romain Forest (THALES), Alberto Blanco, Oriol Farràs, Jordi Ribes, David Sánchez (URV)</td>
</tr>
<tr>
<td><strong>Partner(s) Contributing</strong></td>
<td>AKKA, EURECOM, MTI, OFFIS, THALES, URV</td>
</tr>
<tr>
<td><strong>Document ID</strong></td>
<td>CLARUS-D4.1-Architecture V1</td>
</tr>
</tbody>
</table>
Abstract

This document describes the first specification of the architecture of the CLARUS framework.
Disclaimer

CLARUS (G.A. 644024) is a Research and Innovation Actions project funded by the EU Framework Programme for Research and Innovation Horizon 2020. This document contains information on CLARUS core activities, findings and outcomes. Any reference to content in this document should clearly indicate the authors, source, organisation and publication date. The content of this publication is the sole responsibility of the CLARUS consortium and cannot be considered to reflect the views of the European Commission.
Table of Contents

1 Introduction ................................................................................................................................. 7
   1.1 Scope of the Document ........................................................................................................ 7
   1.2 Outline .................................................................................................................................. 7
   1.3 Revision History .................................................................................................................. 8
2 Background ..................................................................................................................................... 10
   2.1 CLARUS Scenarios ............................................................................................................. 10
   2.2 Requirements and Use Cases .............................................................................................. 12
3 CLARUS Architectural Approach ............................................................................................... 14
   3.1 Design Implications ............................................................................................................. 14
   3.2 High Level Architecture ...................................................................................................... 15
4 CLARUS Architecture Specification .......................................................................................... 17
   4.1 CLARUS Access .................................................................................................................. 17
      4.1.1 Requirements for the CLARUS Access Process ............................................................ 18
      4.1.2 User Registration Module ............................................................................................ 19
      4.1.3 User Authentication Module ......................................................................................... 20
   4.2 CLARUS Data Operations .................................................................................................... 21
      4.2.1 Privacy-Preserving Data Storage .................................................................................. 21
         4.2.1.1 Storage with Encryption ......................................................................................... 21
         4.2.1.2 Storage with Searchable Encryption (SE) ............................................................... 21
         4.2.1.3 Storage with Homomorphic Encryption (HE) ......................................................... 22
         4.2.1.4 Storage with Data Anonymisation ........................................................................... 23
         4.2.1.5 Storage with Data Splitting ..................................................................................... 24
      4.2.2 Data Update ................................................................................................................... 24
      4.2.3 Data Retrieval ............................................................................................................... 25
         4.2.3.1 Retrieval of Encrypted Data .................................................................................... 25
         4.2.3.2 Retrieval with Anonymised Data .............................................................................. 25
         4.2.3.3 Retrieval with Split Data ........................................................................................ 26
4.2.4 Privacy- Preserving Search........................................................................................................ 27
  4.2.4.1 Search over Encrypted Data .............................................................................................. 27
  4.2.4.2 Search over Anonymised Data .......................................................................................... 28
  4.2.4.3 Search over Split Data.................................................................................................... 28
4.2.5 Privacy Preserving Computation ............................................................................................. 29
  4.2.5.1 Computation over Encrypted Data .................................................................................. 29
  4.2.5.2 Computation over Anonymised Data .............................................................................. 30
  4.2.5.3 Computation over Split Data .......................................................................................... 30
4.3 CLARUS Monitoring & Administration ....................................................................................... 32
  4.3.1 Administration ................................................................................................................... 32
    4.3.1.1 Configuration .................................................................................................................. 33
    4.3.1.2 Interactions with User Interface: ...................................................................................... 34
    4.3.1.3 Integration and Interoperability with the IT Infrastructure: ........................................... 34
    4.3.1.4 Interface with other CLARUS Modules ........................................................................ 34
  4.3.2 Security Policy Management .................................................................................................. 34
    4.3.2.1 Overview ...................................................................................................................... 35
    4.3.2.2 Data ............................................................................................................................. 37
    4.3.2.3 Interface with other CLARUS modules ........................................................................ 37
    4.3.2.4 Technical needs ............................................................................................................. 37
  4.3.3 Monitoring ............................................................................................................................. 38
    4.3.3.1 Monitoring Components ............................................................................................... 39
    4.3.3.2 Interaction between the Monitoring Module and other CLARUS Modules .................... 40
    4.3.3.3 Installation of the Monitoring Module (the place where it is located) ............................... 41
    4.3.3.4 CLARUS Data/Access/Support for the Monitoring Module .......................................... 41
    4.3.3.5 Additional Information regarding the Monitoring Module ........................................... 41
4.4 CLARUS Access Policy and Key Management ............................................................................. 42
  4.4.1 Intra-Proxy Access Policy Management ............................................................................... 42
    4.4.1.1 Overview ...................................................................................................................... 42
    4.4.1.2 Definition of the User Profiles ....................................................................................... 42
    4.4.1.3 Definition of the User Authorisations on Outsourced Datasets ..................................... 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.1.4</td>
<td>Data</td>
<td>44</td>
</tr>
<tr>
<td>4.4.1.5</td>
<td>Interfaces with other CLARUS Modules</td>
<td>44</td>
</tr>
<tr>
<td>4.4.1.6</td>
<td>Technical Needs</td>
<td>45</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Inter-Proxy Synchronisation</td>
<td>46</td>
</tr>
<tr>
<td>5</td>
<td>CLARUS Interfaces</td>
<td>47</td>
</tr>
<tr>
<td>5.1</td>
<td>USER-CLARUS Protocol Module</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>CLARUS-CSP Protocol Module</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>49</td>
</tr>
<tr>
<td>7</td>
<td>Bibliography</td>
<td>50</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Scope of the Document

This document specifies the first version of the architecture of the CLARUS platform. The main component of this platform is a proxy in charge of protecting customers’ data in a transparent way, while these are stored and processed in the cloud.

The design of the CLARUS architecture is defined based on the outcomes of WP2, namely the description of the use cases in deliverable D2.1 [1] and the list of requirements specified in D2.2 [2]. The architecture described in this document captures the main technical modules to be developed in relation to one another, in order to help the combination of different cloud services such as data storage or search operations with different protection techniques (encryption, anonymisation, data splitting/merging). This document aims at answering the following questions:

- How does CLARUS protect the storage and processing of data in the cloud?
- How is CLARUS configured?
- How is CLARUS protected?
- How does the end-user, or end-user applications, communicate with CLARUS?

With this aim, the document presents:

- the adopted architectural approach;
- a set of CLARUS modules proposed to tackle the challenges of ensuring the security and privacy of the storage and processing of the data in the cloud;
- the technical integration of these various modules.

In the first version of the CLARUS architecture, the document only focuses on an individual CLARUS proxy (even though this proxy may manage several users within the same organisation). In the second and final version (which will be delivered at M15), it is assumed that several CLARUS proxies will further interact with each other in order to implement, for example, collaborative services outsourced to the cloud. Furthermore, while this first version focuses on the problem of privacy preserving storage and processing of the data, the next version of the architecture will also include the specification of some auditing services.

1.2 Outline

This document is structured as follows:

- Section 2 reviews the CLARUS scenarios and summarises the requirements that motivate the development of the CLARUS framework.
• Section 3 discusses the architectural approach and proposes a high level architecture of the CLARUS proxy.
• The different CLARUS modules are specified in Section 4. These modules are grouped in four main building blocks that focus on the secure data operations, the access to CLARUS, the access policy and key management, and the monitoring and administration of the proxy.
• Section 5 describes two additional modules which correspond to the interface between the end-user and CLARUS on the one hand, and the cloud and CLARUS on the other hand.

1.3 Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>22/07/2015</td>
<td>EURC</td>
<td>Initial ToC</td>
</tr>
<tr>
<td>0.2</td>
<td>31/07/2015</td>
<td>EURC &amp; OFFIS</td>
<td>Second version of ToC</td>
</tr>
<tr>
<td>0.3</td>
<td>25/08/2015</td>
<td>AKKA &amp; EURC &amp; MTI &amp; OFFIS</td>
<td>Initial input from partners on different modules and background</td>
</tr>
<tr>
<td>0.4</td>
<td>27/08/2015</td>
<td>AKKA &amp; EURC &amp; MTI &amp; OFFIS</td>
<td>Partners update their input following the reviews</td>
</tr>
<tr>
<td>0.5</td>
<td>02/09/2015</td>
<td>AKKA &amp; EURC &amp; MTI &amp; OFFIS</td>
<td>Partners add and update figures describing the modules Second version of the architecture</td>
</tr>
<tr>
<td>0.6</td>
<td>04/09/2015</td>
<td>AKKA &amp; EURC &amp; MTI &amp; OFFIS &amp; URV</td>
<td>New ToC, URV modules included, AKKA figures included, OFFIS figures included</td>
</tr>
<tr>
<td>0.7</td>
<td>17/09/2015</td>
<td>AKKA &amp; EURC &amp; OFFIS &amp; URV</td>
<td>New ToC, URV modules included, AKKA figures included, OFFIS figures included</td>
</tr>
<tr>
<td>0.8</td>
<td>23/09/2015</td>
<td>AKKA &amp; EURC &amp; OFFIS &amp; URV</td>
<td>Several updates and comments</td>
</tr>
</tbody>
</table>

© CLARUS Consortium 8 / 50
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Contributors</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>30/09/2015</td>
<td>AKKA&amp;EURC&amp;OFFIS&amp;URV&amp;THAL</td>
<td>Several updates and new contributions</td>
</tr>
<tr>
<td>1.0</td>
<td>08/10/2015</td>
<td>AKKA&amp;EURC&amp;MTI&amp;OFFIS&amp;URV&amp;THAL</td>
<td>Final versions of sections</td>
</tr>
<tr>
<td>1.05</td>
<td>19/10/2015</td>
<td>RHUL&amp;URV</td>
<td>Review of v1.0</td>
</tr>
<tr>
<td>1.1</td>
<td>26/10/2015</td>
<td>EURC</td>
<td>Addressed most of the comments from RHUL and URV</td>
</tr>
<tr>
<td>1.2</td>
<td>27/10/2015</td>
<td>AKKA&amp;EURC&amp;OFFIS&amp;URV</td>
<td>Final updates</td>
</tr>
<tr>
<td>2.0</td>
<td>29/10/2015</td>
<td>EURC</td>
<td>Final version and release</td>
</tr>
</tbody>
</table>
2 Background

2.1 CLARUS Scenarios

The newly proposed CLARUS platform aims at protecting cloud customers’ data while they are stored and processed at the cloud. This platform consists in a proxy located between customers and the cloud, which protects all the data exchanged between them.

CLARUS scenarios include different numbers of cloud customers and different numbers of cloud providers. In a case where one user connects to a single cloud, no user management would be actually needed. However, since in a realistic use case and under real world conditions, many users will usually want to connect to one or several cloud providers, the architecture must fulfil their needs. In short, the architecture must provide the ability to process the following scenarios:

1 Cloud Customer connects to 1 Cloud Service Provider

![Figure 1: One user (u) connects to a single CSP through CLARUS](image)

As illustrated in Figure 1, in the simplest scenario, one user connects to a single cloud provider through the CLARUS proxy.

n Cloud Customers connect to 1 Cloud Service Provider

Figure 2 shows the case where two users connect to a single CSP through CLARUS.
Multiple users (two in this example) connect to a single CSP

Cloud Customers connect to Cloud Service Providers

It is often desirable to store data splits in different locations so that no single provider can ever access the entire data set. This scenario is shown in Figure 3, where two users connect to two different accounts or CSPs. CLARUS handles the connection management and data splitting and also stores the information about which data set is stored at which CSP. For all users, the storage and processing of data are entirely transparent, meaning that end-users will not even know whether the data are split or whether they are stored at a single CSP. The CSP will also be unaware of how data is being split, so that it could only access to partial views of the whole data.

Multiple users connect to multiple CSPs or CSP accounts
There may be reasons for one user to support the connection to multiple CSPs, for example to fortify data security by using data splitting. However, this scenario will absolutely be transparent to the user, so he will not be able to detect any difference in the usage of CLARUS, and the scenario is therefore comparable to the one described in Figure 3.

2.2 Requirements and Use Cases

The collecting of requirements and their specification address the foremost need for the definition of the global architecture of the expected CLARUS solution and the need for its design. This requirement gathering has been done in D2.2 [2] (WP2) and has enabled the statement of both functional and non-functional requirements. This list of requirements have been further completed with some additional non-functional requirements based upon legal and ethical constraints (cf. D2.4 [3]), as well as with standardisation considerations (cf. D2.5 [4]). In WP2, we agreed on a common representation of the requirements:

- Requirements are uniquely identified. This ensures traceability of the requirements in future phases of the project.
- Requirements are prioritised following the MoSCoW [5] scale.

The full set of requirements enunciated in WP2 deliverables should constitute the input keys for the architectural approach and the definition of the fundamental features of CLARUS, which are the aims of this deliverable. Indeed, the final solution must conform to the constraints set by the statement of these requirements. Therefore, the main key points that emerged during the collecting of requirements must be kept in mind: these are recalled hereafter.

The requirements collected in D2.2 [2] are directly inherited from the specification of the use cases, based upon the different possible uses of CLARUS by the different types of actors. Namely, we considered two main categories of actors:

- Actors involved in use cases related to infrastructure management (e.g. IT or development teams, ie. administrator, etc.):
  - Installation (on premises or in the cloud) and configuration of the CLARUS solution use cases;
  - Monitoring use cases.

- Actors embodied by the end-users of the CLARUS solution, involved in use cases of:
  - Access control and security policy management use cases, driven by the security manager of the organisation.
  - Management and processing of the outsourced dataset use cases, as a data provider (i.e. owner) or a data consumer.

From the perspective of data management, which reflects the core of the CLARUS tool, use cases may be classified in four scenarios:

- data storage;
• search in the cloud;
• update in the cloud;
• computation in the cloud.

Each of these scenarios can be further defined by the following data management operations to be supported by CLARUS: ‘create’, ‘retrieve by id’, ‘update by id’, ‘search by attributes’, ‘update by attributes’ and ‘compute’. These primary functionalities can be assimilated to the ‘building blocks’ that must be part of the application layer in terms of architecture, on which CLARUS security and privacy features have to be grafted.

The access-control use case, which sets the authorisations of users in existing outsourced datasets, requires the CLARUS architecture to include some user registration and authentication mechanisms. The inter-proxy synchronisation, which allows interaction between several CLARUS proxies, should also be considered.

The security policy management use case leads to a proposal of security policies that rely on a notion of 'dataspace' (virtual space making the link between the cloud storage services and end-user applications) involved in the protection mechanisms. Thus, outsourced datasets should be protected according to their usage, taking into account:

• the expected parameters depending on the protection solution;
• the supported protocols;
• the appropriate endpoints of the cloud storage services;
• the supported types of data and the attributes to be protected.

The optimal choice of the security mechanism to apply in each particular case will depend on these factors. Therefore, the design of CLARUS needs to take these factors into consideration. This use case thereby reveals additional features that are supposed to be part of the design of the CLARUS tool.

The monitoring use case imposes to design monitoring features in a broad way by detecting and preventing intrusion attempts either from the end-user premises, or from the CLARUS proxy or at the cloud service provider level. The functional behaviour outlined delivers appropriate counter-measures in an ‘Attack Tolerant System’ mode, resulting in service continuity after an intrusion warning. A web-based output synthesising all metrics retrieved and attack management logs should also be designed.

Installation and configuration use cases set the needs for the implementation of the required protection techniques, the different modules, and the administration of the CLARUS proxy.

All these key concepts, emanating from all the different use cases, have been listed in WP2 under the form of concise, detailed, testable and traceable descriptions. They constitute, in addition to the basic legal aspects and the standardisation roadmap, the set of requirements that now must be an integral part in the design of a CLARUS solution compliant with the targeted outcomes.
3 CLARUS Architectural Approach

3.1 Design Implications

Based on the different CLARUS scenarios and the requirements, we present the design alternatives for the CLARUS platform and their implications on the architecture. As the goal of this project is to provide the cloud customer with some means to ensure the protection of the data outsourced to the cloud and the operations applied to them, the proposed solution is to define a gateway (proxy) which the exchanges between cloud customers and cloud service providers go through. Notably, CLARUS is supposed to prepare and secure data prior to outsourcing. CLARUS is also presumed to translate cloud customers’ queries (cloud service providers’ responses respectively) for the cloud service provider (for the cloud customer respectively).

The most straightforward manner to implement CLARUS is to have one proxy for all cloud customers, that is, having CLARUS as a centralised entity that provides security services to various registered cloud customers. The main limitation of this approach is that instead of trusting the cloud as it is the current case, one now has to place his trust in the centralised CLARUS server which, similarly to the cloud, cannot be governed by the cloud customer policy, and whose compliance cannot be verified easily.

The ideal case from a security point of view is to have one CLARUS proxy per end-user. Here we should differentiate between the end-user and the cloud customer: Namely, a company acting as a cloud customer will have as many CLARUS proxies as employees, since the employees are the actual end-users of the cloud service. Besides the important financial cost of such approach, there is the issue of synchronisation between different proxies belonging to the same organisation, because users of the organisation are likely to interact and access to common datasets. To further illustrate this limitation, take the case where an end-user $u$ within an organisation uploads the data to the cloud using her CLARUS proxy. Once the data generated by user $u$ is uploaded, it is desirable to let other users access the data using their own proxies. This entails that those proxies should possess the right keying material and metadata to be able to strip the layers of security protection off the accessed data. The easiest way to enable this process is to have proxies, belonging to the same organisation, store their keying material and metadata in a centralised server. These sensitive data can be further protected using techniques, such as attribute-based encryption, which ensure that only authorised users can access a certain key or metadata. The fallout of such design choice is that proxies form a single logical entity albeit physically independent. Although a decentralised approach is preferable, it will naturally rely on more involved cryptographic techniques and it will increase the communication cost between the proxies drastically. This may deter organisations from opting for CLARUS as the incurred cost in terms of bandwidth and computation cannot be overlooked.

This is why in this project, we make a design choice that allows each cloud customer to run a single CLARUS proxy on its premises. Considering the case of a company, this design choice implies that all the employees in this company will use the same CLARUS proxy to upload the sensitive data they generate. While in this manner, we do not need any proxy synchronisation within the organisation, we still have to find novel techniques to enable
communication and sometimes synchronisation between proxies belonging to different legal entities that want to share/exchange some of their data. Recall that this aspect will be covered in the second version of this document.

In the following, we provide a high level overview of the CLARUS proxy that outlines the modules that will constitute CLARUS and, as such, help improving the security in the cloud.

### 3.2 High Level Architecture

We provide an overview of the architecture of the CLARUS proxy, which, as introduced in the previous section, will be in charge of protecting the storage and processing cloud customers’ data.

![Figure 4 CLARUS proxy architecture V1](image-url)

The architecture of the proposed CLARUS proxy is illustrated in Figure 4. Because the main goal of the CLARUS proxy is to provide a set of mechanisms to protect the security and privacy of cloud customers’ data, the core building block of the architecture, named as “Data Operations”, encapsulates a set of privacy modules for...
different cloud services (data storage, data retrieval, search, computation) and protection techniques (encryption, data splitting and anonymisation). These modules will mainly implement the privacy preserving techniques designed in WP3. This core building block naturally requires the support of the “Access Policy and Key Management” building block, which will help each privacy module protect outsourced by defining suitable access policies and providing appropriate security materials that will be stored in dedicated Access Policy DB and Key Store respectively. The bootstrap and further administration of the proxy will be controlled by different modules regrouped in the “Monitoring and Administration” building block: while the Administration module is in charge of adding and configuring different modules and repositories, the Security Policy Management module manages the security policies defining different protection rules based on the data type and the underlying data/communication protocols for each dataspace; additionally, the Monitoring module will ensure that the framework is intrusion tolerant. Finally, since several end-users will connect to the same CLARUS proxy, the “CLARUS Access” block will control their accesses and identities with the help of the User-Registration and User-Authentication modules.

The architecture also defines two modules, namely the USER-CLARUS and the CSP-CLARUS Protocol modules, which serve as interfaces for the end-user and the cloud, respectively.

The first version of this architecture only focuses on the existence of one CLARUS proxy and does not therefore tackle the problem of inter-proxy synchronisation, which is left for the second version of the architecture. Furthermore, the second version of the architecture will also include a set of security auditing and integrity solutions that will enhance trust in the cloud environment.
4 CLARUS Architecture Specification

4.1 CLARUS Access

This building block basically provides the initial and permanent access for CLARUS users to the CLARUS proxy and, thus, to the entire CLARUS solution. This access includes the initial authentication requests as well as permanent working sessions through the application.

The application the end-user works with, connects to the CLARUS proxy instead of the original service the application would usually use to store its data. For this process, the CLARUS proxy provides a special module called the USER-CLARUS Protocol Module, which is explained in detail in Section 5.1. In short, this module includes specific plug-ins for each supported protocol, which are in charge of the data interception, parsing and processing so that the data and commands can be forwarded with a homogeneous format to the following modules. It is important to state that the processing of the input data depends on the protocol and the individual application that should be able to connect to CLARUS. As an example, the following situation can be considered.

An application uses an SQL database to store and query its data; the connection is established through a network and the application directly connects to the database server. If the application had to be used with CLARUS, the network server in the application would be changed to connect to the CLARUS proxy instead. The proxy then offers a module (USER-CLARUS Protocol module) that inspects the incoming data and picks out the data that should be secured while handing over the rest of the data directly to a corresponding storage system, normally also in the cloud.

To be able to match a user session with the incoming data stream, the data stream is inspected for user identification traits, for example SQL login phrases, HTTP tokens or similar unique identification elements. CLARUS may also rely on authentication services already existing within the company, for example, based on Single Sign on. If the user session does not exist yet, CLARUS implicitly authenticates the user, provided that the identification traits are adequate for that. If not, a previous explicit user authentication of that user to CLARUS is needed. Once a user is authenticated, a user session is created for that user. By using this technique, the CLARUS proxy can identify the user without any impact on the application use.

The Protocol module prepares the data according to the defined configuration and forwards the data to the corresponding CLARUS modules that take care of the encryption, anonymisation and other methods that are appropriate for the kind of protection and data operations to be supported over the data.

Even though this version of the document focuses in the one proxy setting we envisage that user profiles can be imported from one proxy to another, as described in D2.2, section 3.4.2.3. This would be done by the User Registration module.
4.1.1 Requirements for the CLARUS Access Process

In this section, the requirements for the access and authentication process are formulated. To provide an overview of the CLARUS access, Figure 5 shows the access patterns of users to the CLARUS proxy. In general, there are several authentication steps during the entire usage process of CLARUS, some of them are explicit and some are implicit. The following three authentication levels are performed in the CLARUS context:

1. User authentication at the CLARUS proxy: A user must be authenticated if he wants to work with the CLARUS proxy. The details of this process are described in Section 4.1.3.
2. Mapping of the incoming data to the CLARUS proxy to an existing user session: The data that is sent by the user must be matched to the existing user session. In special cases, if the application protocol allows it, the authentication can be performed based on the application data. If the user session does not exist, the session may be created via implicit authentication.
3. CLARUS proxy access to each connected CSP: The CLARUS proxy holds all login information (security credentials) to the connected CSPs. Therefore, no specific user accounts exist at the CSP for the users that connect to CLARUS. Instead, CLARUS manages the connections to the CSPs and forwards all user data to them applying the data protection operations, if needed.

The following requirements apply to these access processes:
• A user who wants to access CLARUS must have a user account at the CLARUS proxy. This account should preferably be backed by the company’s user management system. However, in a first version, an embedded user management system can be implemented.

• Access to the CLARUS proxy should be as transparent as possible for the user, although authentication may be required.

• If possible, CLARUS integrates the enterprise’s authentication mechanism such as Single Sign On (SSO), provided that the implementation of the enterprise’s authentication mechanism relies on open standard protocols (e.g. SAML, OAuth2, OpenID, etc.).

• If CLARUS cannot integrate the enterprise’s authentication mechanism (or there is no enterprise’s authentication mechanism), the User Authentication module should provide an embedded authentication mechanism.

• New user accounts are created by the security manager with a default user profile. This is done through the “User Registration Module”.

• Whenever a user connects to CLARUS either explicitly or implicitly through the client application, this user must be fully authenticated. If this authentication cannot be guaranteed by the data protocol alone, a prior explicit authentication is needed: in this case, CLARUS manages a “User Database” where user accounts, as well as possible registration and profile data are stored. In both cases, the process is performed by the “User Authentication Module”, which either needs an explicit authentication or performs the authentication based on protocol data.

• Both the “User Authentication Module” and the “User Registration Module” offer a web-based interface which can be used by any modern browser.

These requirements guarantee that each user that connects to the local CLARUS proxy, which is running in a trusted zone, is authenticated and accesses the system as a trusted user of the company allowed by the security manager. CLARUS will then take care of the trust towards the cloud providers and, possibly, towards other CLARUS proxies. However, this latter case is described in version 2 of this document.

4.1.2 User Registration Module

The “User Registration Module” is mainly used by the security manager to create new users. The module offers a web interface that can be used by any modern browser. The security manager will then create or import new user profiles and store default profiles for these users. The structure of the module is shown in Figure 6.
4.1.3 User Authentication Module

The “User Authentication Module” offers, on the one hand, a web interface for each CLARUS user, which the latter can use to authenticate himself if there is no SSO mechanism available in the company. A user session is then created and used by the protocol module to map the incoming application data to a user account. On the other hand, the module also contains interfaces that allow authentication through the application protocol data so that the authentication process becomes totally transparent to the user. The structure of the module is shown in Figure 7.
4.2 CLARUS Data Operations

In this section, we describe the modules of the “Data Operations” building block, which handle the protection of data exchanges between the end-user(s) and the cloud(s). The generic operations that need to be protected by CLARUS are data storage, data retrieval, data update, search, and computation.

4.2.1 Privacy-Preserving Data Storage

4.2.1.1 Storage with Encryption

The Storage with Encryption module, illustrated in Figure 8, takes as input the data to be encrypted and uploaded and the access policy that specifies which party can retrieve it from the CSP. The module generates an appropriate encryption key, encrypts the data with it and further stores this encryption key in the key store with the policy. We assume that this module uses symmetric encryption and the decryption key is therefore equal to the encryption key and will therefore be used for further data retrieval queries. Finally, the module transfers the encrypted data to the cloud through the CLARUS-CSP Protocol module.

4.2.1.2 Storage with Searchable Encryption (SE)

The Storage with Searchable Encryption (SE) module takes as input the data to be uploaded (which could be a text file or rows of a database) and the access policy that specifies what authorised parties it can search. Namely, the organisation to which the data belongs may define access policies (using the Intra-Proxy Policy and Key Management Module) that discriminate between its employees based on their roles in the organisation. Provided with the access policy and the data, the module fetches from the User Key Store the secret keys that match the policy. Next, the module first selects keywords that best represent the data, and then builds a secure searchable index of these keywords after their obfuscation. Note that the keyword obfuscation and the index construction follow the access policy: indeed, the module prepares the index and obfuscates the keywords in...
such a way that it distinguishes between the capabilities of authorised parties (e.g. a part of the index can be searched by any employee in the organisation, whereas the rest is only accessible to the organisation managers). Afterwards, the module produces a set of search keys that will empower end-users to search parts of the uploaded data in accordance with their credentials. These search keys are stored in the Data Key Store together with the data identifier. Finally, the module stores the metadata produced during the index construction in the Metadata Database, and transfers the data to the encryption module and the index to the cloud. Figure 9 illustrates the Storage with SE module.

![Figure 9 Storage with Searchable Encryption Module](image)

4.2.1.3 Storage with Homomorphic Encryption (HE)

The Storage with Homomorphic Encryption (HE) module, illustrated in Figure 10, takes as input the data to be uploaded, the uploader’s access policy and the description of the operation (or operations) that should be supported by the cloud server. Provided with the description of the operation, the module picks the most suitable encryption scheme, that is, the encryption that requires the least computational effort and keeps the communication overhead to a minimum, and stores the metadata needed to carry out the encryption and decryption correctly in the metadata database. Then, based on the access policy, it contacts the User Key Store to retrieve the encryption keys that best translate the access policy. Finally, the encryption module encodes the data into the encryption plaintext space, encrypts it and sends it to the cloud.
4.2.1.4 Storage with Data Anonymisation

The Storage with Data Anonymisation module, also described in Figure 11, receives from the USER-CLARUS Protocol module the clear data to be stored. Then, it checks within the Security Policy the expected accuracy (e.g., precision of decimal numbers, degree of data coarsening) and/or the anonymisation criterion (e.g., k-anonymity level [6]) and the specific data within the dataspace to be protected (e.g., rows in a structured database) defined by the security manager for the corresponding dataspace during the configuration stage. A data anonymisation process is then carried out on the data in coherence with the specified parameters. The anonymised outcome is finally forwarded to the CLARUS-CSP Protocol module to be stored in the cloud. Note that anonymisation is irreversible. Thus, from this moment onwards, queries and computations over these anonymised data will only provide coarsened results. Consequently, the module does not need to store any metadata associated to the anonymisation process and queries could be also performed directly to the cloud.

Figure 11 Storage with Data Anonymisation Module
4.2.1.5 Storage with Data Splitting

The Storage with Data Splitting module receives from the USER-CLARUS Protocol module the clear data to be stored. As shown in Figure 12, it further retrieves from the Security Policy the set of cloud locations (with the required access credentials) that have been specified by the security manager as available to store data. It also retrieves the privacy rules defined over the data to be protected (i.e., what pieces of data can be stored in clear within the same location without incurring in a privacy threat). According to these rules, a data splitting process is performed, so that data are split in chunks of clear data, each one fulfilling with the privacy criteria. The process would minimise the number of cloud locations to use, even though a minimum number of locations would be needed and should be available for certain data according to the privacy rules. After this, the splitting criteria, that is, how the data have been split and in which cloud each chunk has been stored, are stored as metadata, and the different data chunks are forwarded to the CLARUS-CSP Protocol module to be independently stored in different clouds.

![Figure 12 Storage with Data Splitting Module](image)

4.2.2 Data Update

Data updates on protected (e.g., encrypted) data can be trivially solved by i) retrieving the whole data from the cloud, ii) unprotecting it, iii) performing the update operation on CLARUS premises and iv) call the dedicated Storage Module (Storage with SE, Storage with Anonymisation, Storage with HE, Storage with Encryption). The only exception is data anonymisation, for which the applied data coarsening may be incompatible with the updating operation. In general, update operations would be not supported with data anonymisation.

On the other hand, updates on split data can be performed more efficiently and directly on the cloud. To do so, the module retrieves from the Metadata DB the splitting rules associated to the data to be updated (i.e., how data has been partitioned in chunks and the cloud location of each chunk). Then, several partial update queries are created, one for each of the data chunks that the update operation involves. Each partial update query is finally sent to the cloud corresponding to the data chunk through the CLARUS-CSP Protocol module. Figure 13 illustrates the Update module for the case where data is split.
Figure 13 Update on Split Data Module

4.2.3 Data Retrieval

4.2.3.1 Retrieval of Encrypted Data

Upon receiving a data retrieval query together with the credentials of the querier, this module, also illustrated in Figure 14, checks the access policy; if the received credentials comply with the corresponding policy, the module retrieves the corresponding decryption key from the key store and the encrypted data from the cloud. The encrypted data is decrypted and the cleartext data are returned to the querier.

Figure 14 Retrieval of Encrypted Data Module

4.2.3.2 Retrieval with Anonymised Data

Since data anonymisation is irreversible, there is no need to perform any “de-anonymisation” process of the results. The end-user will always receive a coarsened version of the data, according to the anonymisation process that has been applied. Thus, as shown in Figure 15, CLARUS will just forward the retrieval query to the cloud. The retrieved anonymised data are finally forwarded to the end-user. If a per-user access level on the data has been
defined by the security manager, an additional coarsening process may be applied over the already anonymised data by the system prior to sending the result to the end-user.

Notice that, due to the straightforward nature of retrieval from anonymised data, in which the retrieved data need not be processed in any way, this retrieval can be also asked directly by the end-user to the cloud (i.e., without using CLARUS). In this last case, however, no access control will be implemented, so that this would be only valid if the data should be accessible for external users.

### 4.2.3.3 Retrieval with Split Data

The retrieval query is processed in order to obtain the splitting rules (splitting criteria and locations of data chunks) that were applied to the data from the Metadata DB. Then, as many queries as chunks involved in the retrieval query are created and individually forwarded to the cloud in which each chunk is stored via the CLARUS-CSP Protocol module. As a result, several individual chunks are retrieved (containing partial but clear data), which are aggregated by inverting the splitting process. The complete data are finally returned to the USER-CLARUS Protocol module. Figure 16 illustrates the Retrieval of Split Data module.
4.2.4 Privacy-Preserving Search

4.2.4.1 Search over Encrypted Data

This module is dedicated to the data stored using the Storage with Searchable Encryption (SE) Module.

Query generation

As shown in Figure 17, on inputs of a search query and the credentials of the querier, the module first gets the access policy, the metadata and the search keys matching the data the querier is interested in. Then, it combines the retrieved information together with the querier’s credentials to generate a search token. The resulting search token is later used to obfuscate and translate the search query for the cloud.

Response generation

Upon reception of the obfuscated search query, the cloud identifies the secure index corresponding to the data to be searched, and proceeds with the computation of the search result using a dedicated cloud-side CLARUS Search over Encrypted module. This cloud-side module is illustrated in Figure 18.
Response processing

On receiving the cloud’s response, the CLARUS Search over Encrypted Data module de-obfuscates the response and derives the result of the search query in plaintext.

4.2.4.2 Search over Anonymised Data

Searching over anonymised data according to a structural criterion is equivalent to data retrieval. The process is straightforward: the search query is directly sent to the cloud and the retrieved results (which are anonymised) are provided to the end-user.

However, searching according to an attribute value or a keyword would be only possible if these refer to a piece of data that has not been anonymised. In general, since data anonymisation is irreversible, we can assume that keyword search over coarsened data would not be supported.

4.2.4.3 Search over Split Data

Searching over split data is essentially very similar to data retrieval. As shown in Figure 19, the query is processed and the metadata related to the split data to which the query refers to are retrieved. Then, the search query is replicated as many times as chunks are involved in the data to be searched. Each resulting query is individually forwarded to the cloud in which each chunk is stored (through the protocol module), which will execute it transparently. A set of partial results are retrieved from the different clouds (again through the protocol module), which are aggregated in coherence with the splitting rules. The final complete result is returned to the protocol module.

Figure 19 Search over Split Data Module
4.2.5 Privacy Preserving Computation

4.2.5.1 Computation over Encrypted Data

This module is dedicated to the data stored using the Storage with HE Module.

Query generation

As illustrated in Figure 20, upon receiving a computation query, the module parses and transforms the computation query in accordance with the homomorphic encryption implemented at the cloud. Upon receipt of the computation query, the cloud performs the requested computation.

Response generation

Thanks to the homomorphic properties of the encryption schemes used to hide the data, the cloud server will be able to perform the required operations without accessing the plaintext data.

Response processing

Provided the querier’s decryption key from the User Key Store and the encryption type from the Metadata DB, the module decrypts the cloud’s response. If the querier is authorised to access the result of the computation, then, its decryption key will enable the module to derive the result of the computation in plaintext, otherwise, the decryption will fail with a high probability. Figure 21 illustrates this phase of the module.
4.2.5.2 Computation over Anonymised Data

Due to the irreversible nature of data coarsening, computation over anonymised data would likely produce also coarsened or approximated results. Only for some very specific operations (e.g., computation of averages) and anonymisation mechanisms (e.g., k-anonymous data microaggregation), can the accuracy of the result be preserved. The process is however straightforward; as shown in Figure 22, once the end-user’s permissions are checked, the computation query is directly forwarded to the cloud, which will directly apply it to the anonymised stored data in a transparent way. The (approximated) result of the operation that is retrieved from the cloud is finally forwarded to the end-user.

![Figure 22 Compute over Anonymised Data Module](image)

4.2.5.3 Computation over Split Data

Depending on the kind of computation to be performed, its enforcement over split data would be more or less complex. For computations that can be individually performed over each chunk of data (e.g., average calculation of individual attributes), the aggregation of the final result is straightforward. For computations that rely on covariances (e.g. correlations) between attributes or data pieces that may belong to different chunks, the computation and aggregation process may require several iterations of partial calculations and a relatively complex integration. Thus, according to the computation query and the fact that it affects data of a single chunk or data spread through different chunks, different strategies will be implemented to split the computation query into several individual queries. These “adapted” computation queries will be then forwarded to the clouds in which the corresponding chunks are stored (via the protocol modules), which will execute them transparently. The partial results retrieved from the different clouds will be finally aggregated in coherence with the mathematical nature of the requested computation and the splitting rules. Figure 23 illustrates the Compute over Split Data module.
Figure 23 Compute over Split Data Module
4.3 CLARUS Monitoring & Administration

In this section, we describe the Administration, Monitoring and Security Policy Management Modules which constitute the Monitoring & Administration Building Block.

4.3.1 Administration

The CLARUS proxy will be provided as a package and with an installation guide, so that to facilitate the deployment of the CLARUS proxy by the IT Team. The IT Team will only need to open an administrator session (the system will control the access rights) on the server where the CLARUS proxy will be executed and then follow the instructions of the installation guide. The IT Team will also be able to upgrade the CLARUS Proxy if needed, by following the upgrade instructions from the installation guide of the CLARUS proxy’s newer version.

CLARUS will provide an installation guide for cloud service providers in order to help them integrate CLARUS in their cloud platforms. Cloud service providers will be able to install one (or more) CLARUS component(s) (tool(s), service(s), library(ies), etc.) by following the instructions of the CSP installation guide. They will also be able to upgrade CLARUS component(s) by following the instructions of the CSP installation guide corresponding to the version they need.

The Administration module, illustrated in Figure 24, aims to help the IT team to manage the CLARUS proxy on the local (trusted) IT system. The IT team will be able to configure the User Repository, the User Authentication module and the access to CSPs; they will also be able to configure the failover mode.

![Figure 24 Administration Module](image-url)
CLARUS will offer the IT Team the possibility to implement, add and remove CLARUS modules. The IT development team could implement modules in order to allow the CLARUS proxy to support features not natively supported (e.g. a protocol used to transfer data or a type of file used to store the data). Then, the IT Team will be able to add or remove those modules previously implemented via a web interface.

CLARUS will provide a web interface with modules (User Repository Web Configuration, Authentication Web Configuration, etc.) associated to each configurable mechanism like user authentication, user repository, cloud storage service, etc. Those modules (User Repository Web Configuration, Authentication Web Configuration, etc.) will be the graphic part required to easily configure each mechanism of the CLARUS proxy.

4.3.1.1 Configuration

User repository

The IT team can configure the CLARUS proxy to connect it to a User Repository for the management of users’ access rights. Two kinds of User Repository will be supported:

- IT user repository delivered by the organisation to which the CLARUS proxy could integrate to search for users and their information;
- An embedded user repository.

The user repository configuration will be stored in the Admin DB.

User authentication

The IT team can configure the authentication mechanism of the CLARUS proxy. The authentication mechanism configuration will be stored in the Admin DB.

Cloud storage (access to CSPs)

The IT team can configure access to the cloud storage service of CSPs. They will be able to set CSPs endpoints (URL) and credentials, and they will be able to control whether one or more CSP(s) are accessible.

Failover mode

The IT team can configure the CLARUS proxy to be part of a failover cluster of CLARUS proxies. The configuration of the failover mode will be stored in the Admin DB.

Implemented Module(s)

The IT development team can implement module(s) in order to allow the CLARUS proxy to support features not natively supported. The IT team can install one (or more) CLARUS module(s) within the CLARUS proxy. Configuration of these (those) implemented module(s) will be stored in the Admin DB.
4.3.1.2 Interactions with User Interface

The IT team is provided with a web interface to manage the CSP.

4.3.1.3 Integration and Interoperability with the IT Infrastructure

The module is designed to support external authentication and authorisation mechanisms that rely on open-standards such as SSO, OAuth, SAML or OpenID.

4.3.1.4 Interface with other CLARUS Modules

The Administration module will interact with other CLARUS modules:

User registration

The Administration module will interact with the User Registration module in order to administrate and configure the User Repository. It will populate the Admin DB that will be used by the User Registration module in order to decide which repository it should use. The Administration module will have the possibility to reload configuration on the user registration mechanism.

User authentication

The Administration module will interact with the User Authentication module in order to administrate and configure the user authentication mechanism. It will populate the Admin DB that will be used by the User Authentication module. The Administration module will have the possibility to reload configuration on the user authentication mechanism.

Cloud storage

The Administration module will interact with the cloud storage service in order to configure CSPs. It will populate the Admin DB that will be used by cloud storage service in order to set access to CSPs. The Administration module will have the possibility to reload configuration on the cloud storage service.

Implemented modules

The Administration module will interact with all modules developed and implemented, which are not natively supported by the CLARUS proxy. It will populate the Admin DB that will be used to load, add, or remove implemented module(s).

4.3.2 Security Policy Management

CLARUS aims to allow end-users to retain control over the stored data in the cloud. The system is designed to support any kind of storage/processing service involved in the IT applications. However, defining how to protect the data requires advanced knowledge in the field of security, which is not obvious for uninitiated people. End users want their data to be protected, but do not know (and usually do not understand) how the data are protected.
In CLARUS, the management of the security policies allows to define how to protect the outsourced datasets. Because this is a critical task that requires knowledge in the field of security, only the security manager is allowed to manage the security policies. However, even for security experts, it could be difficult to define the best security strategy. The system will assist and guide the security manager in the definition of a security policy, because the kind of data protection that could be offered will be likely restricted by the expected data usage and the accuracy of the outcomes. It also allows the end-users to view how their data are protected in a simplified way.

In order to enhance trust in the system, end-users are allowed to retain control of the authorisations on their stored data.

The Security Policy Management module, illustrated in Figure 25, allows defining the users’ profiles, their rights and their access levels on the outsourced datasets. The security manager is allowed to manage all the user profiles and all the user authorisations defined in the system whereas end-users are allowed to manage only the user authorisations on their own outsourced datasets.

The management of the security policies and the management of the users’ profiles and of the authorisations enable a set of complementary and coherent security rules that the CLARUS proxy can apply to protect the outsourced datasets.

4.3.2.1 Overview

The module in charge of the management of the security policies describes the storage/processing service(s) involved in the IT applications used by the end-users, and it also defines what (and how) to protect in the associated outsourced dataset(s).

The module focuses on the description of the service layer of the IT applications, so that the CLARUS proxy can set up a gateway to intercept traffic related to the outsourced dataset(s). The module also focuses on the definition of the security strategy to apply in order to protect the outsourced dataset(s).

Description of the service layer of the IT applications

In order to support the storage/processing service(s) involved in the IT applications, the system must support the protocol(s) and must provide dedicated endpoint(s).

The module allows selecting the protocol(s) involved in the IT applications.

For each selected protocol, the internal endpoint must then be configured (e.g. listen port). Obviously, the endpoints of the storage/processing service(s) running on the cloud must also be configured (e.g. URLs). In that way, the system will act as a gateway between the applications/services running on premises and the storage/processing service(s) running on the cloud.
**Setting up the security policy**

Once the service layer of the IT applications has been described, the strategy to protect the data must be defined.

**Description of the IT application’s data**

The system must be aware of the structure of the data to be transferred. It relies on the ‘Protocol’ plug-ins defined in the Protocol modules described in Section 5, but if the protocol is not dedicated to a type of data and it is able to transfer any kind of data (e.g. data are as black boxes for S3), the data type(s) must be described by the ‘Data Type’ plug-ins. The module allows selecting the type(s) of data that constitute an outsourced dataset.

**Select the data to protect**

Depending on the IT applications, not all data need to be protected. Usually, only data related to privacy and confidentiality need to be protected. The module allows selecting the attributes to protect.

**Data usage definition and refinement**

Defining the data usage is essential because not all security techniques are compatible with all data usages (and expected accuracies). Defining the data usage has a direct impact on the security strategy. The module assists the security manager, by allowing him defining the data usage, and then pre-selecting the most appropriate combinations of the security techniques according to his needs. If more than one combination is suggested, the security manager has to select one.
Additionally to the data usage definition, the module also allows to refine configuration of each security technique involved in the security strategy. For instance, the module allows defining the default outcome’s accuracy on the outsourced datasets, which is a parameter for data anonymisation (i.e., the degree of data coarsening/ generalization to apply). The data anonymisation process will be carried out on the data in coherence with the specified parameters.

4.3.2.2 Data

The management of the security policies requires the configuration of the dataspace with the following attributes:

- **Storage/processing services**: the list of protocols with their internal endpoints and the associated endpoints of the storage/processing services running on the cloud;
- **Data types**: the list of data types that can be protected. Data types either are deduced from the protocol(s) or are explicitly defined;
- **Protection needs**: the list of data attributes to protect, the data usage and the default outcome accuracy;
- **Security policy**: the combination of security techniques to apply so as to protect the outsourced dataset(s), with their configuration refined.

The data related to the management of the security policies are critical because they define how to implement a gateway and how to protect the outsourced dataset(s). The system will fail if those data are lost or corrupted.

4.3.2.3 Interface with other CLARUS modules

The module in charge of the management of the security policies interacts with other CLARUS modules:

- All modules in the ‘Data Operations’ building block;
- The two CLARUS Protocol Modules, which rely on Protocol and Data Type plug-ins.

4.3.2.4 Technical needs

The security manager is provided with a web interface to manage the dataspaces. The end-users are provided with a web interface to see/check/visualise how their data are protected.

**Presentation Layer**

A web server runs the web interface of the module in charge of the management of the security policies, accessed by the security manager.

**Business Layer**

An application server runs the module in charge of the management of the security policies.

Note: the application server must not run the ‘Data Operations’ modules. Loose coupling ensures that the ‘Data Operations’ modules are not impacted if the application server dedicated to Security Policy Management crashes. In case of failure or of maintenance tasks, the latter can be independently restarted.
Persistence Layer

A storage back end (e.g. database) stores the data related to the management of the security policies.

The back end provides fast access in read-only mode so that the CLARUS proxy can implement a gateway as quickly as possible.

The back end provides replication mechanisms in order to preserve data quality.

The data disaster recovery procedure of the system must ensure that those data can be restored in case of failure.

4.3.3 Monitoring

Monitoring is a solution required to ensure the correct operation of the whole system, by continuously analysing the data flow and information exchanges to quickly detect any security and privacy issue that might compromise the overall system security and data integrity.

The Monitoring module in CLARUS is linked to the Administration module and they work together to guarantee that the system works properly according to the established performance and security policies.

In CLARUS, the Monitoring module observes in runtime input and output events without disturbing the normal operation of the system, and analyses this information to detect security threats, privacy issues, attacks, and suspicious behaviours that might put the security of the system at risk. In addition, the Monitoring module integrated in CLARUS is able to decide the best strategy to repel the security issue, and to trigger countermeasures to let the system continue working properly, without being affected by the detected security threat.

The Monitoring module (see Figure 26) is composed of a monitoring server installed in the CLARUS proxy, and distributed probes. The distributed probes can be installed in the CLARUS proxies, in the cloud services interacting with it, and in intermediate network points. These probes sniff network traffic that is analysed by the monitoring server to check that security properties are fulfilled. The monitoring server includes a database to store the sniffed data (monitoring database), and another database with security information and rules to detect security issues (security database). The monitoring server receives the data from the probes, and correlates the information to detect security incidents that may put the integrity of the information in the system at risk. When an attack is detected, automatic countermeasures are triggered in order to mitigate its effects and to allow the normal operation of the system without interruption.
The activity of the Monitoring module is transparent to the end-user, while the detected, avoided and/or mitigated security issues are reported to the administrator and/or to the security manager to take further actions if necessary.

4.3.3.1 Monitoring Components

1. Data collection
Data collection is performed by the data probes installed in the CLARUS proxies, in the cloud services they access, and at different points of the network premises infrastructure. The captured data is sent in real-time to the monitoring server that stores it in the monitoring database. This captured data includes application logs, database communications, network traffic (in case of encrypted traffic, at least the request source, request frequency, and timestamps are analysed), etc.

2. Data analysis
The data received from the distributed probes is analysed in the monitoring server. For that, the Security database is consulted to check security properties, security rules, and detection of unexpected behaviours. There is also the possibility of accessing external databases (accessed through secured communications) containing additional security properties, security checking information, and global threat reports.

3. Security faults' detection
The data analysis performed in the monitoring server when correlating the information coming from various probes, and by following the security information stored in the security and security policies databases, may
eventually detect an attack or any other kind of security issue. In that case, complete information about the attack nature (source of the attack, timestamp, hosts involved, suspicious actions detected, etc.) will be collected to select the most appropriate reaction and/or mitigation strategy to allow the system to continue running with the minimal disturb.

Note that collaboration between several monitoring servers is required in order to detect distributed attacks.

4. Reaction/mitigation strategies

Each kind of attack will require specific countermeasures to eliminate or mitigate it. For that, the information regarding the attack extracted by the monitoring server will constitute the basis to decide the action to perform (e.g., block source of the attack, close incoming port, etc.). Once the correction measure has been executed, a completed report will be sent to the administrator, and any new security information will be stored in the monitoring and security databases for future reference. In case of not being able to automatically stop the attack or its consequences, an alert will be sent to the administrator and/or to the security manager (depending on the detected attack), requesting the action to be performed.

5. Visualisation

In order to inform the involved parties of an attack, or simply to keep them informed about statistics from the monitoring service, we can distinguish two cases depending on the actors:

- Administrator and security manager: alerts and mitigation reports will be sent to the administrator and/or security managers to request actions in case of a non-manageable attack, and to inform him/her about the repelled attacks.
- End users: statistical information can be offered to the end-users regarding the utilisation of their data, such as what, when, and who has accessed their information.

4.3.3.2 Interaction between the Monitoring Module and other CLARUS Modules

The CLARUS proxy will incorporate a Monitoring module containing a monitoring database, a security database, one or several probes to collect data, and the means to process the gathered data to detect and mitigate security incidents, and to inform the involved actors.

The Monitoring module can be fully configured to detect diverse kinds of attacks and unexpected/suspicious behaviour. It needs access to the list of authorised users (through the user authentication module; it can also support interoperable authentication mechanisms such as a Federated ID framework) and other security-related information that will be provided by the administrator. Security rules to trigger alerts and countermeasures will also be configurable by the administrator, thus offering flexibility in the configuration of the monitoring service.

A web interface, integrated in the monitoring module, will enable performing the required configuration and specifying attack reaction activities.
4.3.3 Installation of the Monitoring Module (the place where it is located)

The monitoring module is composed of a series of components:

- The monitoring server: installed in the CLARUS proxy.
- The monitoring database: included in the monitoring server to store collected data used to detect security incidents.
- Security database: generally located in the monitoring server, it contains security checking information and rules to detect attacks and unexpected behaviour, as well as mitigation mechanisms. Additional security databases (external to CLARUS) with complementary security information can also be consulted. For accessing this kind of databases, secure communications are required to avoid additional security risks.
- Probes: to collect data coming from diverse sources. They may be distributed i.e., installed in diverse parts of the system: the CLARUS proxy, databases, cloud services, end-user and network sides.

4.3.4 CLARUS Data/Access/Support for the Monitoring Module

Different components of the monitoring module require different execution and access rights:

- Monitoring server: It requires execution privileges and computation power to process all the gathered information. In addition, full access to the gathered data, and read/write access to the databases are needed. Furthermore, appropriate rights to execute the required countermeasures will allow repelling and/or mitigating the detected security issues. Secure communications should be granted to inform the involved parties of the actions performed, as well as to access external databases.
- Probes: they need read access rights to the data they are meant to collect. Depending on where the data are located (application, network, etc.), different rights must be granted to the probes. Secure communication capabilities between probes and the monitoring server are also required.

4.3.5 Additional Information regarding the Monitoring Module

The monitoring processes are designed to be transparent to the end-user, although they can be informed about statistical information if authorised by the administrator.

Apart from receiving security reports and alerts, the administrator will be allowed to configure, add, and eliminate security rules, to add new sources of information and new mitigation mechanisms, to visualise monitoring reports, and in general, to fully configure the monitoring module to fulfil the required security policies.

As mentioned above, the detection of coordinated attacks requires the collaboration of several monitoring modules. The monitoring structure presented in this document will evolve with the specification of the second version of the CLARUS architecture to enable the operation of distributed monitoring modules, installed in various CLARUS proxies, and the possibility of detecting coordinated attacks.
4.4 CLARUS Access Policy and Key Management

This section describes the Access Policy and Key Management building block, which manages the security materials and access policies and distributes them to the different modules in the “Data Operations” building block.

4.4.1 Intra-Proxy Access Policy Management

4.4.1.1 Overview

This module allows defining the access rights of the end-users on the storage/processing service(s) involved in the IT applications and supported by the system. The module also allows to define the permissions and, optionally and if applicable, access levels of the end-users on the outsourced dataset(s) associated to the storage/processing service(s). Based on the access policy, the module also stores the relevant encryption keys or other security materials to the key store. Figure 27 illustrates this Intra-Proxy Access Policy Management module.

![Diagram of Intra-Proxy Access Policy Management Module]

4.4.1.2 Definition of the User Profiles

The module focuses on the management of the user profiles. A user profile defines the access rights a user has on the storage/processing service(s) involved in the IT applications. The Access Policy module controls access to
the service(s) according to the user profile. A user profile also contains dedicated security materials (e.g. keys) involved in the access (including creation) to the outsourced dataset(s) within the storage/processing service(s) involved in the IT applications.

Access rights

The module manages the access rights of the users. The access rights are either none, read-only or read-write:

- a user with no access rights cannot use the service(s) and so cannot access the existing outsourced dataset(s) associated to the service(s);
- a user with read-only access rights can access the existing outsourced dataset(s) associated to the service(s) in read-only mode and according to his/her permissions on the associated outsourced dataset(s) (see below);
- a user with read-write access rights can access the existing outsourced dataset(s) associated to the service(s) in read-write mode and according to his/her permissions on the associated outsourced dataset(s) (see below);
- a user with read-write access rights can also create outsourced dataset(s) associated to the service(s). A user is the owner of the outsourced dataset(s) he/she has created, and so he/she is allowed to modify permissions of the other users on his/her outsourced dataset(s) (see below).

Security materials

The module allows managing the security materials (including encryption or search keys) dedicated to the users. A user has dedicated security materials that are built by the system according to the security policy and according to the access rights of the user. Security materials are all necessary resources which are involved in the access to the data (create, read, write, delete) by the user using the storage/processing service(s).

4.4.1.3 Definition of the User Authorisations on Outsourced Datasets

The module also focuses on the management of the authorisations on the outsourced dataset(s). User authorisations are the permissions and the access levels a user has on the outsourced dataset(s). The CLARUS proxy controls access to the outsourced dataset(s) according to the user permissions and adjusts the outcomes according to the access level of the user.

Permissions

The module manages the permissions of the users on the existing outsourced datasets. The permissions are either read, update and delete:

- read permission allows a user to read an existing outsourced dataset, provided that the service(s) used to access the outsourced dataset implements some kind of read operation (which is almost always the case);
• update permission allows a user to modify an existing outsourced dataset, provided that the service(s) used to access the outsourced dataset implement some kind of write operation;
• delete permission allows a user to delete an existing outsourced dataset, provided that the service(s) used to access the outsourced dataset implement some kind of delete operation.

Note that a user that creates an outsourced dataset has all permissions (read, update, delete) and is also allowed to modify the permissions of the other users on the new outsourced dataset. Obviously, the security manager is allowed to modify the permissions of all end-users on all existing outsourced datasets.

**Access level**

The module allows changing the outcome accuracies of the end-users on the existing outsourced datasets.

The access level is a degree of coarsening/generalisation or a percentage of the outcome’s accuracy an end-user has when read accessing an outsourced dataset. Necessarily, this degree will be equal to or below the coarsening applied to the data outsourced to the cloud (i.e., through an anonymisation mechanism).

**4.4.1.4 Data**

The management of the user profiles and of the authorisations implies storing the following data:

• **User profile:**
  o **Access rights:** the list of access rights (read, write) an end-user has on each storage/processing service;
  o **Security materials:** the list of the resources involved in the access to the outsourced dataset(s) by the end-user for each storage/processing service.

• **User authorisations:**
  o **Permissions:** the list of permissions (read, update, delete) an end-user has on each outsourced dataset;
  o **Access levels:** the list of access levels an end-user has on each outsourced dataset.

**Sensitivity and criticality**

The access rights and permissions are critical data. The outsourced dataset(s) could not be accessible if access rights or permissions are lost or corrupted.

Also, outcome accuracies are critical data. The outcomes could be false if outcome accuracies are lost or corrupted.

Finally, security materials are critical data because they are involved in the access to the outsourced dataset(s). If security materials are lost or corrupted, the outsourced dataset(s) may not be usable.

**4.4.1.5 Interfaces with other CLARUS Modules**

The module interacts with the following CLARUS modules:
- The ‘Security Policy Management’ module: to manage the security materials dedicated to the users according to their profile
- The ‘Data Operations modules’: the user profiles and the authorisations generated by the module are read by the ‘Data Operations modules’ in order to control the access to the storage/processing service(s) and the access to the associated outsourced dataset(s).

### 4.4.1.6 Technical Needs

The security manager is provided with a web interface to manage the user profiles.

The security manager and the end-users are provided with a web interface to manage the authorisations.

**Presentation layer**

A web server runs the web interface of the module in charge of the management of the user profiles and of the authorisations.

**Business layer**

An application server runs the module in charge of the management of the ‘Access Policy and Key management’ building block.

Note: the application server must not run the ‘Data Operations’ modules. Loose coupling ensures that the ‘Data Operations’ modules are not impacted if the application server dedicated to the access policy management crashes. In case of failure or maintenance tasks, the latter can be restarted.

**Persistence layer**

A storage back end (i.e. ‘Access Policy DB’) stores the user profiles (access rights) and the authorisations (permissions and access levels).

Another storage back end (e.g. file server, keystore) stores the security materials of the users (i.e. ‘Key Store DB’).

Both back ends provide fast access in read-only mode so that the CLARUS proxy can take into account the modifications as quickly as possible.

Both back ends provide replication mechanisms in order to preserve data quality.

Both back ends provide fail-over mechanisms in order to ensure SLA for the CLARUS proxy.

The data disaster recovery procedure of the system must ensure that the back ends data can be restored in case of failure.
4.4.2 Inter-Proxy Synchronisation

This module is in charge of synchronising the key store and the Access Policy DB between several proxies. It will be described in detail in the next version of this deliverable.
5 CLARUS Interfaces

5.1 USER-CLARUS Protocol Module

The USER-CLARUS Protocol Module which is illustrated in Figure 28, serves as the entry point to the CLARUS proxy. The user applications connect to the proxy using a specific protocol/port (e.g. HTTP, SQL), which is supported by one of the Protocol end-point plug-ins. These are in charge of determining the user’s identity, with the support of the User Authentication Module, and establishing the user session. The session information, including the user role, permissions and requested data, is used to locate the appropriate Security Policy, which is forwarded, along with the user commands and data (processed by the appropriate data type plug-in), to the Data Operations Modules. The CLARUS proxy will orchestrate the needed ‘Data Operations’ modules and the CLARUS-CSP Protocol module, by using the provided information. Results retrieved from the cloud that have passed through the Data Operations modules are processed inversely.

For commands which do not require protection or, if desired, for protection mechanisms that are perfectly transparent (i.e., anonymisation), the USER-CLARUS Protocol module will forward the input message to the CLARUS-CSP Protocol module.

Figure 28 USER-CLARUS Protocol Module
5.2 CLARUS-CSP Protocol Module

The CLARUS-CSP Protocol Module serves as the communication interface between the internal CLARUS modules and the CSP. As shown in Figure 29, different plug-ins will be in charge of supporting different CSP APIs (e.g. a REST plug-in, an SQL plug-in, an S3 plug-in, a Dropbox plug-in). This module will receive the commands passed by the ‘Data Operations’ Modules, along with the protected data and the Security Policy, and will locate the appropriate CSP credentials and plug-ins to send the commands to the cloud. Responses from the cloud are processed inversely.

Figure 29 CLARUS-CSP Protocol Module
6 Conclusion

The main objective of this deliverable was to specify the architecture of the CLARUS framework that allows protecting the storage and processing of customers’ data in the cloud. In this aim, the document included the identification of the main architectural requirements by analysing the CLARUS use cases and the requirements defined in WP2. Before focusing on each CLARUS module, the document presented the high level overview of the CLARUS architecture.

Moving forward, this document will evolve to describe the complete CLARUS architecture including the specification of the interaction between several CLARUS proxies and the design of some additional auditing modules.
7 Bibliography


