Secure Provisioning of Cloud Services
based on SLA Management

SPECS Project - Deliverable 5.2.1

Evaluation and lessons learnt from scenario on "Real-time monitoring, reporting and response to security incidents related to a CSP"

Version no. 1.1
19 July 2016

The activities reported in this deliverable are partially supported
by the European Community's Seventh Framework Programme under grant agreement no. 610795.

Deliverable information

Deliverable no.:	D5.2.1
Deliverable title:	Evaluation and lessons learnt from scenario on "Real-time monitoring, reporting and response to security incidents related to a CSP"
Deliverable nature:	Report
Dissemination level:	Public
Contractual delivery:	19 July 2016
Actual delivery date:	19 July 2016
Author(s):	Jolanda Modic (XLAB)
Contributors:	Miha Stopar (XLAB)
Reviewers:	Ruben Trapero (TUDA), Jesus Luna (CSA), Massimiliano Rak (CeRICT)
Task contributing to the deliverable:	T5.2
Total number of pages:	48

Executive summary

The focus of this task is to develop a pilot application, using a real service in a production environment, with which we set up a negotiated cloud service, monitor it, and respond to Cloud Service Provider (CSP) related security incidents. In particular, the goal is to develop an application that validates the *Secure Storage* user story using the real world provider Koofr[1].

Both documents of the task T5.2 (namely D5.2.1 and D5.2.2) present development of the Secure Storage application which integrates (i) the SPECS framework enabling all phases of the SLA life cycle, (ii) a set of security mechanisms offering negotiable security features, and (iii) Koofr providing the cloud storage. Through the proof of concept application, the End-users can negotiate and monitor a cloud storage service, enhanced with specific security features:

- Client-side encryption enforcing *confidentiality* and *integrity*;
- Detection and proof of violations related to *write-serializability* (i.e., consistency among up-dates) and *read-freshness* (i.e., assurance that the requested data is always fresh as of the last update);
- Backup of stored data.

In particular, this document presents:

- Secure Storage application: We summarize the functionalities offered with the Secure Storage application by discussing the Secure Storage user story and associated validation scenarios.
- Application development process: We introduce all artifacts needed for the development of the Secure Storage application. Namely, we define the cloud service provisioned by the application, the security mechanisms required to enforce and monitor the defined cloud service, and security metrics with which End-users negotiate specific configurations of the defined cloud service. Lastly, we present the SLA Template that specifies all described attributes and enables automated negotiation, enforcement, and monitoring of the secure storage service.
- Architecture: We present the design of the Secure Storage application, outlining the role of Koofr and the role of the SPECS framework. We present security mechanisms developed for the Secure Storage application in detail, outlining the offered security features, the way we monitor them, and elaborating on the root cause analysis and remediation actions for security incidents that affect these features.

The technical aspects of the Secure Storage application (i.e., installation, usage, testing) are presented in the final deliverable D5.2.2.

[1] http://koofr.eu/

Table of contents

Index of figures

Index of tables

1. Introduction

Cloud Service Providers (CSPs) offer their services through Service Level Agreements (SLAs) and the importance of SLAs is undisputed. However, most of these SLAs are relatively plain and usually cover at most availability and support. While having these properties specified in SLAs is important, there is a variety of security incidents that can occur and are not covered by these guarantees (CSPs do not offer advanced security metrics/SLOs with which they would be committed to monitor them). For example, imagine the scenario where for some reason the End-user's (End-user's) data on the cloud gets corrupted. In this case, the service is still available, the support is available, too, but most probably the CSP does not have a mechanism to restore the corrupted file. Moreover, it is very likely that the data corruption would only be detected by the End-user when trying to access the file for the next time (which might be months after the corruption date). Thus, there is a need to introduce new security properties in SLAs and develop new mechanisms to enforce and monitor them.

In this document, which is describing the Secure Storage application, we focus on security properties that are most critical in the cloud storage domain, i.e., *confidentiality*, *integrity*, *write-serializability* (i.e., consistency among up-dates), and *read-freshness* (i.e., assurance that the requested data is always fresh as of the last update). Apart from providing these guarantees to an End-user, there is also a need to *continuously monitor* the system and to *automatically react* in case of detected violations in order to assure that the CSP's committed SLOs are fulfilled. (Note that in SPECS we only monitor what we guarantee in SLAs. Thus every detected system failure or security incident is associated to a violation of at least one signed SLA.)

To this end, the application developed for the purpose of offering secure storage service integrates (i) two security mechanisms that are able to enforce and monitor the above mentioned security properties (negotiated through SLAs), and (ii) the SPECS framework which enables the automated management of SLAs. Moreover, the integrated security mechanisms form a *proof-based system* which (apart from having the functionality to detect SLA violations when they occur, notifying them to the CSC immediately, and remediating them) enables the End-user to *prove a violation* of some commitment to the CSP, and to enable the CSP to *disprove any potential false accusations* from the End-user.

In order to empirically validate the outcomes of this task and to test application's functionalities (monitoring, detection, and remediation of synthetic incidents) in near real-time, we use cloud storage service provided by Koofr.

The remainder of this document is structured as follows. In Section 2 we present other deliverables of the project that provided an input for this task. The user story and associated validation scenarios, which served as a base for developing the Secure Storage application, are discussed in Section 3. The process of developing the application is described in Section 4, and the architecture of the developed application, the role of Koofr, the role of the SPECS framework, and details about involved security mechanisms are presented in Section 5. The document concludes with Section 6, which summarizes the results and reports about the value of the developed application.

2. Relationship with other deliverables

Development of the Secure Storage application relies on the results of many other tasks of the project. The starting points are deliverables of the task T5.1. Deliverables D5.1.1 and D5.1.2 report the Secure Storage user story and associated validation scenarios, respectively. The application is developed according to the guidelines presented in deliverable D5.1.3.

Deliverables D1.1.3 and D1.3 provide an overview of the framework's architecture and interactions among its components, respectively, and deliverable D1.5.1 offers an insight into the integration process of the SPECS framework and validation applications.

The details about SPECS core modules, which orchestrate the SLA life cycle, are taken from deliverables D2.3.3 (SLA negotiation), D3.4.1 (SLA monitoring), and D4.3.3 (SLA enforcement). Deliverable D4.3.2 provides information about the initial version of security mechanisms integrated with the application.

The results presented in this document provide an input for the complementing deliverable D5.2.2, which presents the technical details of the Secure Storage application, and the deliverable D4.3.3, which describes the final version of the security mechanisms developed for the Secure Storage user story.

Note that in deliverables D5.2.1 and D5.2.2 we present the implementation details associated to the security mechanisms integrated with the Secure Storage application, whereas in deliverables D4.3.2 and D4.3.3 we present their general usage in terms of the SLA life cycle (which metrics mechanisms enforce, how they monitor them, what are the remediation actions in case of their violation, etc.). Hence the bidirectional relationship between deliverables of tasks T4.3 and T5.2.

The security mechanisms developed under task T5.2 also provide feedback to the business validation in WP6 (deliverable D6.2.3).

The discussed relationships are depicted in Figure 1.

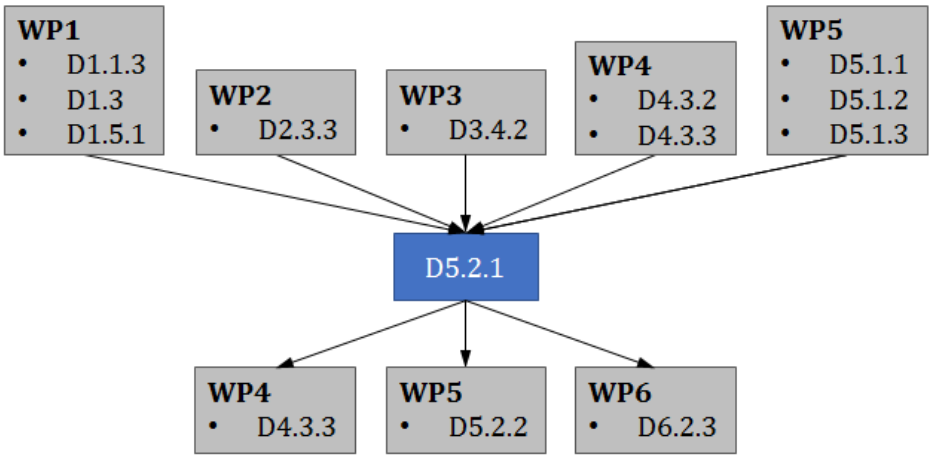

Figure 1. Relationships with other deliverables

3. SPECS Secure Storage application description

The Secure Storage application is one of the validation applications introduced in deliverable D5.1.2, realizing the Secure Storage user story defined in deliverable D5.1.1. The application offers cloud storage enhanced with specific security features to customers through SLAs, and is a part of the SPECS solution portfolio (see D6.2.2 and D6.2.3). As we will discuss in Section 5, we use a real-world cloud storage provider Koofr [7] and the security mechanisms integrated into the application have been developed according to requirements expressed by the Koofr team. The application expands and improves the cloud storage service offered by Koofr by enforcing the client-side encryption and enabling the monitoring of some advanced security properties (integrity, write-serializability, read-freshness).

In the following subsections we present the user story and validation scenarios for the application.

3.1. User story: Secure Storage

The Secure Storage user story (denoted as SST) describes a non-expert End-user, who wants to store data with a remote cloud provider. The End-user wants confidentiality guarantees, such that a provider itself cannot access the stored data.

On top of the required confidentiality guarantees, the Secure Storage service developed in SPECS ensures the following:

- *Cloud storage with backup* guaranteeing business continuity through means of disaster recovery;
- *End-2-end encryption* enforcing confidentiality and integrity of the stored data.
- *Detection and proof of violations related to write-serializability* (i.e., consistency among updates);
- *Detection and proof of violations related to read-freshness* (i.e., requested data always being fresh as of the last update).

Note that, apart from the cloud storage, Koofr does not natively provide other features mentioned above.

For further details about the Secure Storage user story see D5.1.1 and Annex B of D1.2.

3.2. Validation scenarios

This section reports validation scenarios specified according to the Secure Storage user story. Initially, validation scenarios were defined in deliverable D5.1.1. Due to refinements in implementation and integration activities, validation scenarios were refined in the second year of the project and reported in deliverable D5.1.2.

There are five validation scenarios associated to the Secure Storage user story:

- SST.1: An End-user aims at acquiring cloud storage with an external provider. The validation scenario presents the SLA negotiation phase during which the End-user negotiates security features of the secure storage service.
- SST.2: An End-user aims at acquiring cloud storage with an external provider, requiring the client-side encryption functionality. The validation scenario presents the SLA negotiation phase during which the End-user negotiates security features of the secure storage service. Moreover, the scenario also includes the SLA implementation

and SLA monitoring phases, which comprise of deploying and configuring mechanisms able to enforce and monitor the negotiated service.

- SST.3: An End-user aims at acquiring cloud storage with a known cloud provider (for which the End-user provides credentials), requiring the client-side encryption functionality. The validation scenario presents the SLA negotiation phase during which the End-user negotiates security features of the secure storage service. Moreover, the scenario also includes the SLA implementation and SLA monitoring phases, which comprise of deploying and configuring mechanisms able to enforce and monitor the negotiated service.
- SST.4: This validation scenario extends validation scenario SST.2, where an SLA alert is raised due to unavailability of one of the servers. The scenario also includes a successful remediation.
- SST.5: This validation scenario extends validation scenario SST.2, where an SLA violation is raised due to unavailability of one of the servers. The scenario also includes a notification to the End-user and an unsuccessful remediation.

In the following table we present the impact of the Secure Storage application on the execution KPIs (defined in deliverable D5.1.1). For each key concern defined in deliverable D5.1.1 we report the percentage of its coverage by the Secure Storage validation scenarios.

Key concern ID	Key concern	Secure Storage application	All SPECS applications
EC_U	User	20%	66.7%
EC_{TS}	Target services	28.6%	100%
EC_{IC}	Invocation chain	66.7%	100%
EC_{SLA}	SLA life cycle	50%	78.9%
EC_{SS}	SPECS services	43.7%[2]	100%

Table 1. Impact of the Secure Storage application on execution KPIs

In the remainder of this section we report the described validation scenarios in whole. Note that although the defined validation scenarios do not cover all possible combinations of the cloud service provided through the application, nor do they cover all possible cyber incidents. However, these validation scenarios provide a base for the integration activities. Based on the defined validation scenarios we define a set of integration scenarios with which we verify correctness of implementation/integration of involved artifacts. The unit and the integration tests executed for the mechanisms, core components, and the application itself cover as much code and situations, that we can conclude that even combinations of the cloud service and associated security incidents/system failure that are not covered by the defined validation scenarios below would be managed correctly.

3.2.1. SST.1 Secure_Storage_Selection

General Information		
ID	*SST.1 - Secure_Storage_Selection*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*

[2] There are 128 requirements associated to the validation scenarios of the Secure Storage user story out of 293 all requirements for the SPECS framework.

Invocation Chain		*IM1-P, IM3*	*Interaction Model 1- SPECS acting the role of Partner*
Scenario Steps			
General Description		*The End-user aims at acquiring a secure storage service from a cloud provider, which fulfils specific security-related requirements. To achieve this service, the End-user negotiates the desired features with SPECS.* *In this validation scenario, the desired features are entirely implemented by an external CSP, while SPECS only provides the End-user with the functionalities to search, rank and select a service, which is compliant to her/his requirements. Moreover, in this scenario, SPECS supports the End-user in signing an SLA with the selected provider.*	
Steps			
1	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application, SPECS Negotiation module*	
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*	
	Trigger		
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*	
	Postconditions		
2	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application*	
	Preconditions		
	Trigger		
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*	
	Postconditions	*A supply chain compliant to the End-user requirements is built.*	
3	Phase	*SLA Negotiation*	
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*	
	Preconditions	*A secure storage service that fulfils the specific security requirements is known to SPECS.*	
	Trigger		
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application. The CSPs also add the cost of each service offer.*	
	Postconditions		
4	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application, SLA Platform*	
	Preconditions	*The End-user shall be logged on SPECS.*	
	Trigger		

	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The service offer is associated with an SLA published by an external CSP. The End-user either:* *1. Accepts and signs the SLA offered by the external CSP;* *2. Does not select any SLA Offer from the list and repeats the whole process from step 1 (possibly specifying a different set of requirements);* *3. Does not select any SLA Offer from the list and exits the application.*
	Postconditions	*In case 1 - the signed SLA is stored by SPECS. The End-user is enabled to invoke the desired service on the external CSP with the configuration information included in the SLA.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_4, SLA_5*	

3.2.2. SST.2 Secure_Storage_Brokering_with_Client_Crypto

General Information		
ID	*SST.2 - Secure_Storage_Brokering_with_Client_Crypto*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs the two capabilities of Database-as-a-Service and End-2-End Encryption in order to detect and prove security-related violations and to locally encrypt her/his data.* *To achieve this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees.* *SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with end-2-end encryption security mechanism. In this scenario, SPECS also provides monitoring functionalities.*	
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	

	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the need of having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service, which fulfils the specific security requirements is not known to SPECS.* *An external CSP offering the Database-as-a-Service compliant with the related End-user's requirements is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Database-as-a-Service is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install, as well as their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	

	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7*	

3.2.3. SST.3 Secure_Storage_with_Defined_CSP

General Information		
ID	*SST.3 - Secure_Storage_with_Defined_CSP*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*The End-user aims at storing encrypted data on a known remote cloud provider which offers a Database-as-a-service. The End-user asks SPECS for End-2-End Encryption capability, needed to locally encrypt her/his data.* *To achieve this service, the End-user also gives SPECS her/his credentials on the chosen provider; SPECS manages these credentials and uses them to log into the chosen provider and store User's data.* *In this scenario, SPECS also provides monitoring functionalities.*	
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	

	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	*The external CSP offering the Database-as-a-Service chosen by the End-user is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the needs of using a specific CSP as Database-as-a-Service provider and having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, the specific CSP defined by the End-user is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service. For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install as well as their configurations.*

6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.* *The credentials of the End-user on the external CSP have been acquired.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module acquires the storage service with the credentials of the End-user on the external CSP and deploys and configures monitoring agents. The SPECS Enforcement module activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7*	

3.2.4. SST.4 Secure_Storage_Brokering_with_Client_Crypto_Alert

General Information		
ID	*SST.4 - Secure_Storage_Brokering_with_Client_Crypto_alert*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		

General Description		*The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs the two capabilities of Database-as-a-Service and End-2-End Encryption in order to detect and prove security-related violations and to locally encrypt her/his data.* *To achieve this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees.* *SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with end-2-end encryption security mechanism. In this scenario, SPECS also provides monitoring functionalities.* *In this scenario, an alert is raised since the Encryption Server component is detected to be down and, since no data are sent from the End-user during the down time, no violation occurs.*
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the need of having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service that fulfils the specific security requirements is not known to SPECS.* *An external CSP offering the Database-as-a-Service compliant with the related End-user's requirements is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Database-as-a-Service is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*

	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	

	Trigger	*The SPECS Monitoring module generates monitoring events due to the deviation of some metrics from set thresholds (since the the Encryption Server component is down).*
	Actions	*The SPECS Enforcement module analyses monitoring events and classifies it as an alert. The root cause of the monitoring event is determined (the Encryption server component is detected to be down, but no data has been sent from the End-user during the down time; thus no violation occurs).*
	Postconditions	*A report on the alert and on the root cause of the monitoring event is created.*
10	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	
	Trigger	
	Actions	*The SPECS Enforcement module reacts by restarting the component before any encrypted files are sent to the server.*
	Postconditions	*The alert is solved.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_10, SLA_11*	

3.2.5. SST.5 Secure_Storage_Brokering_with_Client_Crypto_Violation

General Information		
ID	*SST.5 - Secure_Storage_Brokering_with_Client_Crypto_violation*	
Version	*2.0*	
User Story	*ST0*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs the two capabilities of Database-as-a-Service and End-2-End Encryption in order to detect and prove security-related violations and to locally encrypt her/his data.* *To achieve this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees.* *SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with end-2-end encryption security mechanism. In this scenario, SPECS also provides monitoring functionalities.* *In this scenario, a violation is detected since the Encryption Server component is detected to be down.*	
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	

	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the need of having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service that fulfils the specific security requirements is not known to SPECS.* *An external CSP offering the Database-as-a-Service compliant with the related End-user's requirements is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Database-as-a-Service is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	

	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*End-user, SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	*The End-user has sent files to encrypt to the server while it is down*
	Trigger	*The SPECS Monitoring module generates monitoring events due to the deviation of some metrics from set thresholds (since the Encryption Server component is down).*
	Actions	*The SPECS Enforcement module analyses monitoring events and detects a violation. The root cause analysis of the monitoring event is determined (the Enforcement module determines that the SLA violation occurred due to the Encryption Server component being down).*
	Postconditions	*A report on the violation and on the root cause of the monitoring event is created.*
10	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	
	Trigger	
	Actions	*SPECS notifies the violation to the End-User through the SPECS Application. The SPECS Enforcement module searches for alternatives for the End-user by building new services.*
	Postconditions	*The SLA is no more fulfilled.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_12*

4. Application development process

In order to develop the Secure Storage application, we took and customized the default SPECS application [4], which integrates the SPECS core modules and enables all functionalities needed to negotiate, enforce, and monitor SLAs. Since the default SPECS application already automates the basic part of the cloud service delivery (namely SLA negotiation, enforcement, monitoring), the customization consists of defining, developing, and deploying the mechanisms that are needed to enforce and monitor the features that we want to offer through the application (confidentiality, integrity, write-serializability, and read-freshness).

The application development process (depicted in Figure 2) starts with the definition of the cloud service that will be delivered through the application and the definition of the features that we want to offer for the chosen cloud service. For example, we define the cloud storage service and define the client-side encryption as one of the features offered with the secure storage service.

Figure 2. Application development process

In the next step we identify (and, if needed, develop) the mechanisms able to enforce and monitor the defined cloud service and its features. For the previous example we would need to identify an existing or develop a new mechanism able to provide the client-side encryption.

In Step 3 we summarize all features that are being offered through the application in an SLA Template. For the example at hand we would need to provide a formal description of the secure storage service and all metrics/SLOs that can be negotiated with respect to the client-side encryption feature. This template serves as the base for the SLA negotiation phase (the SLA Template provides the information to the Negotiation module about what can be negotiated and enforced – for details about the negotiation process in SPECS see deliverable D2.3.3).

The last step of the process comprises the deployment of the developed security mechanisms and the created SLA Template, to make them available to the developed application. In practice this means registering any newly developed security mechanisms and associated Chef[3] cookbooks with the SPECS Chef Server (for details see D4.3.2), and registering the created SLA Template with the SLO Manager[4].

In the following subsections we present the details of the first three steps while the last one, which is not application specific, is discussed in deliverable D5.1.3.

Note that the entire application development process is conducted by following the steps defined in deliverable D5.1.3.

[3] In SPECS, the automated deployment and configuration of resources and services is orchestrated with Chef [6]. For further details on the use of Chef in SPECS see deliverable D4.2.2.

[4] Component of the Negotiation module that stores and manages SLA Templates. For details see D2.3.3.

4.1. Cloud service definition

With the application developed in task T5.2, we want to offer *secure storage* service. In particular, we want to offer cloud storage, enhanced with the following features:

- Detection and proof of violations related to *write-serializability* (WS) and *read-freshness* (RF);
- End-2-end encryption enforcing *confidentiality* (CO) and *integrity* (IN);
- Backup which ensures that in case of violations related to WS, RF, CO, and IN the corrupted or missing data can (to some extent) be recovered. By "to some extent" we mean that the database can be restored to the state of the last completed backup. Any data lost or corrupted between the last completed backup and the occurrence of the violation cannot be restored.

Offering a cloud storage with no additional security guarantees covers validation scenario SST.1, whereas implementing all additional security properties offered through the application covers validation scenarios SST.2, SST.3, SST.4, and SST.5.

In the next section we discuss the mechanisms that are integrated with the application in order to enforce and monitor the above mentioned security features.

4.2. Security mechanisms preparation

In order to enforce secure storage with backup and with the functionality to detect and prove violations of WS, RF, and IN, we integrate the **Database and Backup (DBB) security mechanism**, introduced in deliverable D4.3.2. The mechanism enforces the capability of "*surviving incidents that compromise the availability and/or integrity of data stored remotely by providing backup service and the detection of violations associated to write-serializability and read-freshness*".

In order to enforce CO, we integrate the **End-2-End Encryption (E2EE) security mechanism**, introduced in deliverable D4.3.2. The mechanism enforces the capability of "*providing client-side encryption enforcing confidentiality*".

The Chef cookbooks for automated management of both mechanisms are available to the Secure Storage application online [5].

In the next subsection we present the SLA Template created for the SPECS Secure Storage service. The template specifies the cloud storage capabilities, controls, and metrics that can be negotiated through the application.

4.3. SLA Template preparation

As discussed in deliverable D2.3.3, the negotiation process in SPECS is based on the SLA Template which specifies the service that can be delivered, the capabilities that can be implemented on top of it, and the security guarantees that can be enforced and monitored for each capability. In case of the Secure Storage application, the delivered service is *Secure Storage*. To enforce features that are discussed in Section 4.1 and enforced by mechanisms reported in Section 4.2, we define two security capabilities:

- **Database and Backup as a Service capability** enforced by the DBB security mechanism.
- **End-2-End Encryption capability** enforced by the E2EE security mechanism.

For each of these two capabilities we identify a set of security controls (taken from the NIST [8] and the CSA [9] security control frameworks) and security metrics that they implement. We report them in the following table.

Note that due to improvements in the implementation, security metrics for E2EE and DBB have been slightly updated in the last 6 months of the project. The initial definitions are reported in deliverable D4.3.2.

Security capability	Database and Backup as a Service
NIST 800-53r4	• CP-2(4) Contingency plan | Resume all missions / Business functions • CP-2(6) Contingency plan | Alternate processing / Storage site • CP-6(1) Alternate storage site | Separation from primary site • CP-9 Information system backup • CP-9(6) Information system backup | Redundant secondary system • CP-10 Information system recovery and reconstitution • SI-7 Software, firmware, and information integrity • SI-7(1) Software, firmware, and information integrity | Automated notifications of integrity violations • SI-7(5) Software, firmware, and information integrity | Automated response to integrity violations
CSA CCM v3.0	• IVS-02 Infrastructure & Virtualization security | Change detection • BCR-01 Business continuity management & Operational resilience | Business continuity planning • BCR-11 Business continuity management & Operational resilience | Policy • AIS-03 Application & Interface security | Data integrity
Security metrics	• **Write-Serializability (WS)**: This metric ensures the End-user consistency among updates of the stored data. In case of WS violations, the End-user will be notified and the system will be restored to the state of the last completed backup. • **Read-Freshness (RF)**: This metric ensures the End-user that the requested data will always be fresh as of the last update. In case of RF violations, the End-user will be notified and the system will be restored to the state of the last completed backup. • **Integrity (IN)**: This metric ensures the End-user integrity of the stored data.
Security capability	End-2-End Encryption
NIST 800-53r4	• SC-12 Cryptographic key establishment and management • SC-13 Cryptographic protection
CSA CCM v3.0	• EKM-01 Encryption & Key management | Entitlement • EKM-03 Encryption & Key management | Sensitive data protection
Security metrics	• **Confidentiality (CO)**: This metric ensures the End-user confidentiality of the stored data. Confidentiality is enforced with end-2-end encryption provided by the Client component. We guarantee that the Client component is audited and thus (used as is) grants the security of encryption.

Table 2. Capabilities, controls, and metrics offered through the Secure Storage application

The created SLA Template for the Secure Storage application is fully reported in Appendix 1.

5. Architecture

The Secure Storage application demonstrates the use of the SPECS framework and SPECS security mechanisms in the cloud storage domain. The goal of the task T5.2 is to create different situations in terms of security incidents associated to the cloud storage paradigm (to identify possible security incidents for the security properties offered through the application, as defined by the validation scenarios), and test the behaviour of the SPECS framework and mechanisms in those situations. Although the defined validation scenarios only cover two particular security incidents, we implement mechanisms that are able to detect others.

As already mentioned in the introduction of this document, the developed application integrates (i) the SPECS framework which orchestrates the SLA life cycle, (ii) SPECS security mechanisms which enhance security level of the cloud storage service by enforcing and monitoring specific security features, and (iii) a real world CSP, namely Koofr, as the storage provider. In this section, we further elaborate on the roles and functionalities of these artifacts.

In the first following subsection, we briefly present Koofr and then we focus on the architecture of the developed application and functionalities of security mechanisms.

5.1. Koofr

Koofr is a cloud storage provider, enabling management of any data (photos, videos, documents), on any storage (enables connection of Dropbox, Google Drive, Amazon and OneDrive accounts), from any device (iOS, Android, Windows Phone, OS X, Windows, Linux, WebDAV, and more), and on any location. It automatically synchronises the Koofr account with your computer, automatically backs up data from phone, and enables one search for all files through all accounts and all devices. For more details see [7] or deliverable D6.2.1.

In SPECS, in particular in task T5.2, Koofr is used as the CSP. More precisely, Koofr is used as the SPECS Owner which integrates the SPECS framework to enable management of SLAs for the cloud storage services that are enhanced with security features enforced by SPECS security mechanisms. As shown in Figure 3, End-users can acquire cloud storage services (by negotiating services through SLAs) through the SPECS Secure Storage application based on the SPECS framework that is integrated with the Koofr service.

Figure 3. The role of Koofr in the SPECS Secure Storage application

At the beginning of the project, the requirements for the functionalities offered through the secure storage application came from Koofr. And since Koofr does not provide any kind of encryption, one of the main requirements was to integrate a mechanism (E2EE) providing the client-side encryption. Moreover, by integrating the mechanism to automatically manage

storage and backup (DBB), we provide with additional negotiable security assurances, namely detecting, notifying, and remediating violations associated to integrity, write-serializability and read-freshness. The second mechanism has been developed to cover additional requirements given by the Koofr team (to support advanced cloud storage properties and monitor them).

For further information about exploitation activities associated to Koofr, see deliverables D6.2.2 and D6.2.3.

5.2. Integration with SPECS framework

As already mentioned in Section 4.2, the Secure Storage application integrates two security mechanisms, namely DBB and E2EE that offer database with backup and client-side encryption, respectively. In order to enable an automated management of SLAs, the following core modules of the SPECS framework are used by the Secure Storage application:

- *Negotiation module* to orchestrate negotiation of security features provided by DBB and E2EE mechanisms.
- *Enforcement module* to enforce negotiated service and react in case of any detected SLA violations.
- *Monitoring module* to continuously check and observe the status of negotiated SLAs.
- *SLA Platform* to offer support to the above mentioned modules.

The role of the SPECS framework is depicted in Figure 4. For further details about specific processes orchestrated by each module see deliverable D1.1.3.

Figure 4. The role of the SPECS framework in the SPECS Secure Storage application

Note that both security mechanisms, namely E2EE and DBB, are standalone security mechanisms that do not integrate, extend or replace any of the SPECS components. As discussed in D4.3.3, they provide (enforce and monitor) additional security assurances to cloud customers in the scope of the secure storage domain, that can be negotiated by End-users in the SPECS SLA negotiation phase.

Since the task T5.2 is focused on security incidents and their management in the cloud, in the remainder of this section we focus on details of the DBB and E2EE mechanisms, in particular on the monitoring of security features they enforce and remediating security incidents associated to them.

5.2.1. DBB and E2EE security mechanisms

With the E2EE security mechanism we offer to the End-user the functionality to locally encrypt and decrypt the data stored in the cloud (with the *Confidentiality* security metric described in Table 2) which ensures confidentiality. Moreover, with the DBB security mechanism we guarantee to the End-user that all violations associated to the integrity, write-serializability and read-freshness of the stored data will be detected, notified, and remediated (by selecting the *Write-Serializability*, *Read-Freshness*, and *Integrity* security metrics defined in Table 2).

To enable these functionalities, we deploy the following components of both mechanisms (introduced in deliverable D4.3.2 and shown in Figure 5) to the acquired cloud resources (in our case, on resources provided by Koofr):

- **Client** component operates directly on the End-user's machine independently from the SPECS framework and the CSP (the End-user downloads the tool from the web store once the SLA is signed). The Client provides a web interface for uploading and downloading files, and for automatically sending attestations to the Auditor component (attestations are signed messages that accompany each End-user's request and each CSP's response, used to verify fulfilment of WS, RF, and IN commitments; discussed in Section 5.2.2). If the CO is requested, the Client provides the end-2-end encryption/decryption of the files being sent/received to/from the CSP. An End-user can use more than one Client component to access the data.
- **Main Server** and **Backup Server** are the main components (*application servers*) deployed on the primary storage site and the backup, respectively. On the primary storage site, the Main Server oversees all configurations, handles all put/get requests, and orchestrates all associated operations (i.e., writes/reads the data, performs backups, sends attestations to the Client). The server component on the backup site is responsible for backups and restorations.
- **Main DB** and **Backup DB** are the *database servers* deployed on the primary storage site and the backup, respectively.
- **Auditor** monitors fulfilment of WS, RF, and IN commitments by performing auditing, i.e., checking if sequences of put/get attestations (received from the Client) form correct write/read chains.
- **Monitoring Adapter** monitors databases on both storage sites (i.e., monitors availability of both servers, availability of both databases, and completeness of backups/restorations) and verifies the CO commitment by continuously monitoring that the web store maintains the correct (latest) version of the Client code.

To maximise the level of security, we deploy these components on five different virtual machines (VMs) as shown in Figure 5. Their configuration depends on the security metrics chosen by the End-user during the negotiation phase.

During the SLA monitoring phase, the Monitoring Adapter and the Auditor continuously send the collected monitoring data (called *events*) to the Monitoring module which determines whether any of the sent events indicate a possible alert/violation and should therefore be further analysed by the Enforcement module. Whenever the Enforcement module is notified about a possible DBB/E2EE alert/violation, the notified event has to be classified, analysed, and remediated (see deliverable D4.3.2). All WS/RF/IN/CO violations are also notified to End-users through the application.

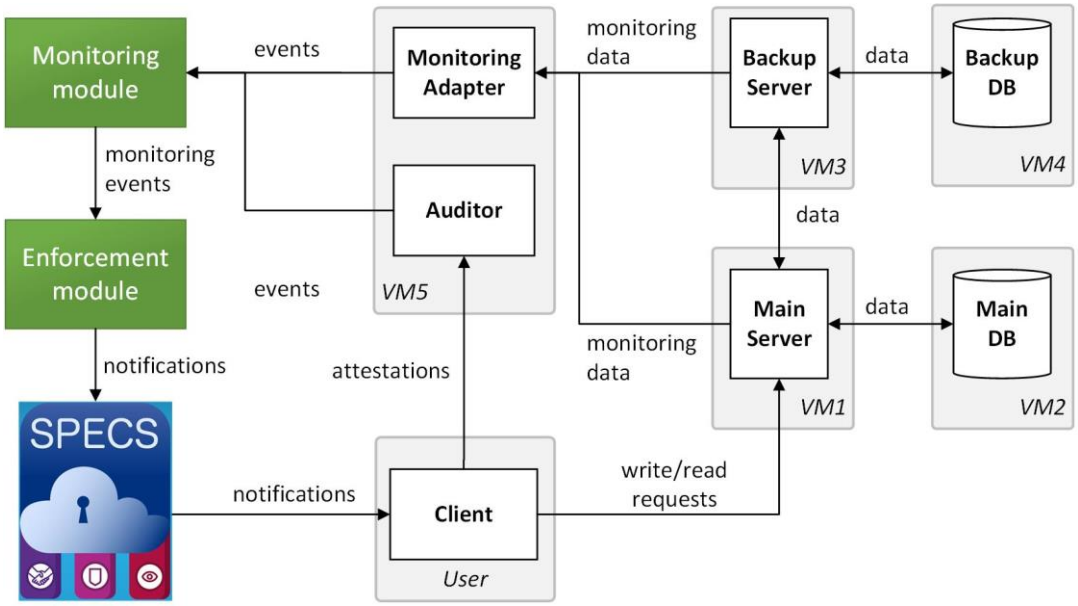

Figure 5. Architecture of the DBB and E2EE mechanisms

In the next subsection, we discuss how the Auditor verifies the fulfilment of commitments related to WS, RF, and IN, and elaborate on how it performs the root cause analysis and decides on the optimal remediation plan. Later we present the monitoring data collected by the Monitoring Adapter.

5.2.2. The Auditor

Attestations are the core objects for the auditing process. They are signed messages that accompany every End-user's request and every CSP's response. The CSP stores End-user's attestations for potential cases when the End-user would trigger false accusations. Similarly, the End-user saves all attestations received from the CSP to prove any violations occurred on the CSP. Additionally, the End-user automatically sends all attestations to the Auditor, which then checks them in order to verify validity of WS, RF, and IN commitments. Once attestations are sent to the Auditor, the Client can delete them.

After each *epoch* (i.e., a predefined fixed period of time) the Auditor checks the chain of attestations for the current epoch. If any errors are detected, the Auditor notifies the Enforcement module that an SLA violation might have occurred.

In the diagrams below we demonstrate the use of attestations. As shown in Figure 6, each time the End-user wants to upload a file to the cloud, the Client performs a `put` request which contains the End-user's data to be stored and a *client put attestation*. The CSP (i.e., the Main Server component) stores the data in the Main DB and returns the *cloud put attestation*. The Client has to provide the client put attestation in order to authorize the overwriting of a certain existing data with a new content. The CSP must respond to the request with the cloud put attestation which affirms that the CSP received the data unchanged and successfully stored it.

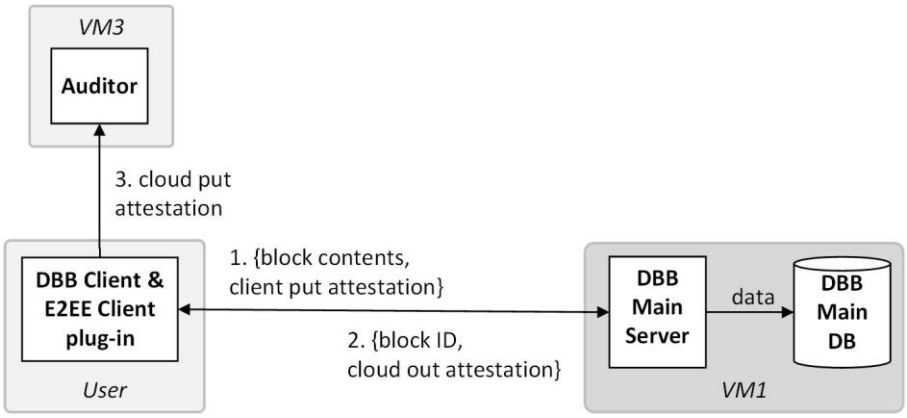

Figure 6. Put attestations

Similarly, as shown in Figure 7, every time the End-user wants to download data from the cloud, the Client performs a `get` request which contains the block ID of the desired data. The CSP (i.e., the Main Server component) returns the requested data along with the *cloud get attestation* with which it certifies that the returned data is the right one.

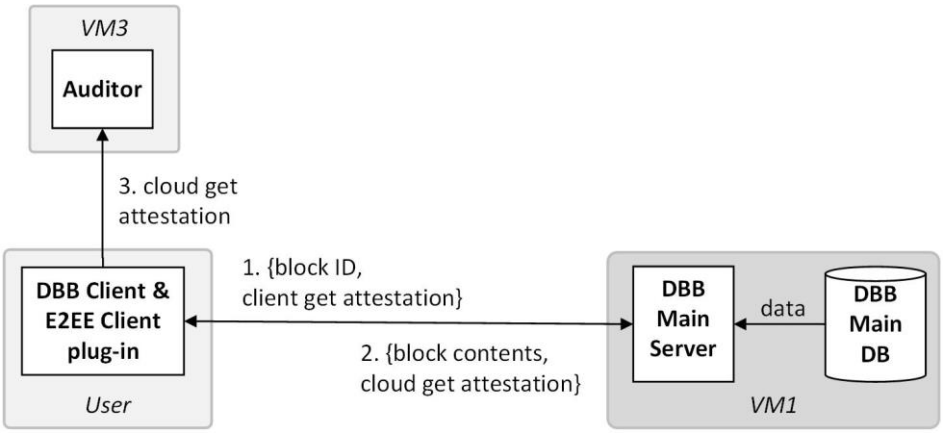

Figure 7. Get attestations

Each attestation is composed of different elements which are hashed and signed. Every attestation includes the *block ID* (i.e., identifier that refers to a block on the cloud) and its *key block version number* (i.e., an integer for a block that increases by one with every update to the block). Depending on the actor (Client vs. Main Server) and the request (`put` vs. `get`), each attestation includes additional fields as seen in Figure 8 and described below. All fields are first concatenated, then a hash is calculated over it, and then the result is signed. For further technical details see [3] (which introduces the idea of the attestation-based cloud storage security properties).

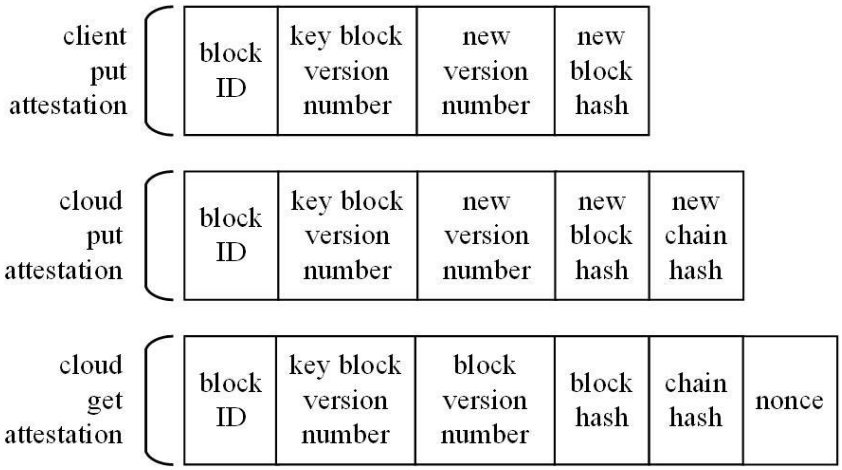

Figure 8. Structure of attestations

The client put attestation additionally contains a *new version number* (which is the old version number, retrieved by get request, increased by 1) and a *new block hash* (hash calculated over the new data). The CSP then takes the previous *chain hash* (i.e., a value updated by each `put` request using the formula *chain hash = hash(attestation data, previous chain hash)*) and new data and calculates the new chain hash. Thus the cloud put attestation contains the same elements as the client put attestation, but has an additional field for *new chain hash*.

The cloud get attestation contains (apart from the block ID and key block version number) the *block version number* (i.e., version number of the block), the *block hash* (i.e., hash calculated over the data at the time when it was stored or at the time when it has to be returned to the Client – depends on implementation), the *chain hash* (calculated over the attestation data and the previous chain hash at the time when it was stored by the CSP in the Main DB), and a *nonce* (a random number given by the Client which is used to prevent the CSP from constructing attestations ahead of time before a certain write happens).

The Client automatically sends all cloud attestations to the Auditor. After each *epoch* (i.e., a predefined fixed period of time) the Auditor checks the chain of attestations for the current epoch. The chain of attestations is correct if for each two consecutive attestations A1 and A2 the chain hash in A2 is equal to the hash of A2's data and A1's hash (if the condition *chain_hash(A2) = hash(data(A2), chain_hash(A1))* holds for two consecutive attestations A1 and A2).

The DBB and E2EE security mechanisms in SPECS are based on the *CloudProof* idea introduced by Popa et al. in [3]. In the remainder of this section we present the initial idea, discuss its drawbacks, and introduce the extended and improved version of the CloudProof system developed in SPECS (for further details see [1]).

The Auditor introduced by Popa et al. is able to detect when End-user's `put` requests are not handled consistently by the CSP (violations of WS or illegitimate modifications of a different nature; however, it does not distinguish between the two). The original Auditor is also able to detect when the End-user has not received the correct data when executing the `get` request (violation of RF). Moreover, the CloudProof's Auditor can detect violations of IN, but only after each epoch, when it checks correctness of chains of all attestations (some `get` request attestation would not have the same data as the `put` request attestation for this data and this

version). We extend the functionalities of the CloudProof by using the fact that the Client can detect such a violation immediately and can trigger the auditing process at that moment to verify it (sends a notification to the Auditor). The Client can detect a violation of IN because it can calculate the hash of the encrypted data and compare it with the value in the attestation (authenticated encryption should be used which means that decryption handled by the Client would report an error when the data is changed illegitimately).

An additional limit of the existing CloudProof's Auditor is that it is unable to determine what the root cause of violations of WS and RF might be. For example, data can be modified by an attacker or by a database error. The latter is outlined with the following scenario. When a `put` request is executed, the server first calculates the chain hash using the received (encrypted) data, returns the attestation (containing the calculated chain hash), and then executes the write operation to the database, which for some reason fails. In this case, the Client would not notice any problems (the chain hash was calculated correctly), but the data stored in the Main DB would not be correct. If there would be a subsequent call on this block, the failure would be noticed (the chain of attestations would be broken), otherwise not. This particular case might lead into a different scenario if the server would first store the data to the database and then retrieve it to calculate the chain hash and return the attestation to the Client. This approach would enable the Client to immediately notice the failure. However, such implementation might not be preferred by a CSP nor by the End-user as it requires additional read operation on the database which will slow-down the entire process.

These remarks show that it is quite difficult to determine the events that lead to a violation as different implementations might cause the failures being detected at slightly different times (we cannot assume that the implementation is not modified by an attacker at some point in time).

Distinguishing between root causes of SLA violations is crucial because entirely different incident responses are required for a system error and an actual attack. While detection of an attacker requires significant changes, like restoring the service on another virtual machine, the detection of a database error might require only switching from a primary database to the backup (followed by a manual inspection of what is wrong with the primary database).

However, it is crucial to be absolutely positive whether the cause of the violation is an attacker or a system error. Below, we explain in what cases we can be sure whether an attacker caused a violation. In some cases we have to admit that the cause cannot be (yet) determined.

It has to be noted that the CloudProof's Auditor by Popa et al. does not take into account violations of IN detected by End-users. The Auditor can by itself detect violation of IN, but not in real-time. The Client can detect in real-time with a `get` request that a block has been illegitimately changed, and send a notification to the Auditor immediately. When the Client sends the `get` request to the server, the cloud get attestation is returned and contains the chain hash and the block hash of the requested data. With received chain hash, the Client can calculate the block hash itself and compare it with the received block hash. If they do not match, the Client detects a violation of IN.

Moreover, not only IN violations can be detected by the Client, a chain hash incorrectness can be discovered as well by checking whether the chain hash returned by the server in the `put`

request is correct. The Client has a chain hash from the last `get` request for a block and can calculate the chain hash that is to be returned by the `put` request for the block (the chain hash that is to be contained in the returned cloud put attestation) using the hash of a block and other block metadata. If the Client's calculated chain hash and the one returned from the Main Server are not equal, the Client detected hash incorrectness. Chain hash incorrectness means a WS violation. It can be detected by the CloudProof's Auditor only at the end of an epoch.

Since all this information detected by the Client is crucial for sustaining the security level specified in an SLA, with our approach all confidentiality and integrity violations, and any detected chain hash incorrectness are immediately notified to the Auditor for further analysis.

In the following we describe an algorithm executed by the Auditor at the end of each epoch and when (if) the Client detects confidentiality and integrity violations or a chain hash incorrectness. The auditing process is also depicted in Figure 9 where each end node outlines metrics that have been affected by the detected violation (in bold), and remediation actions that have to be taken to recover from it.

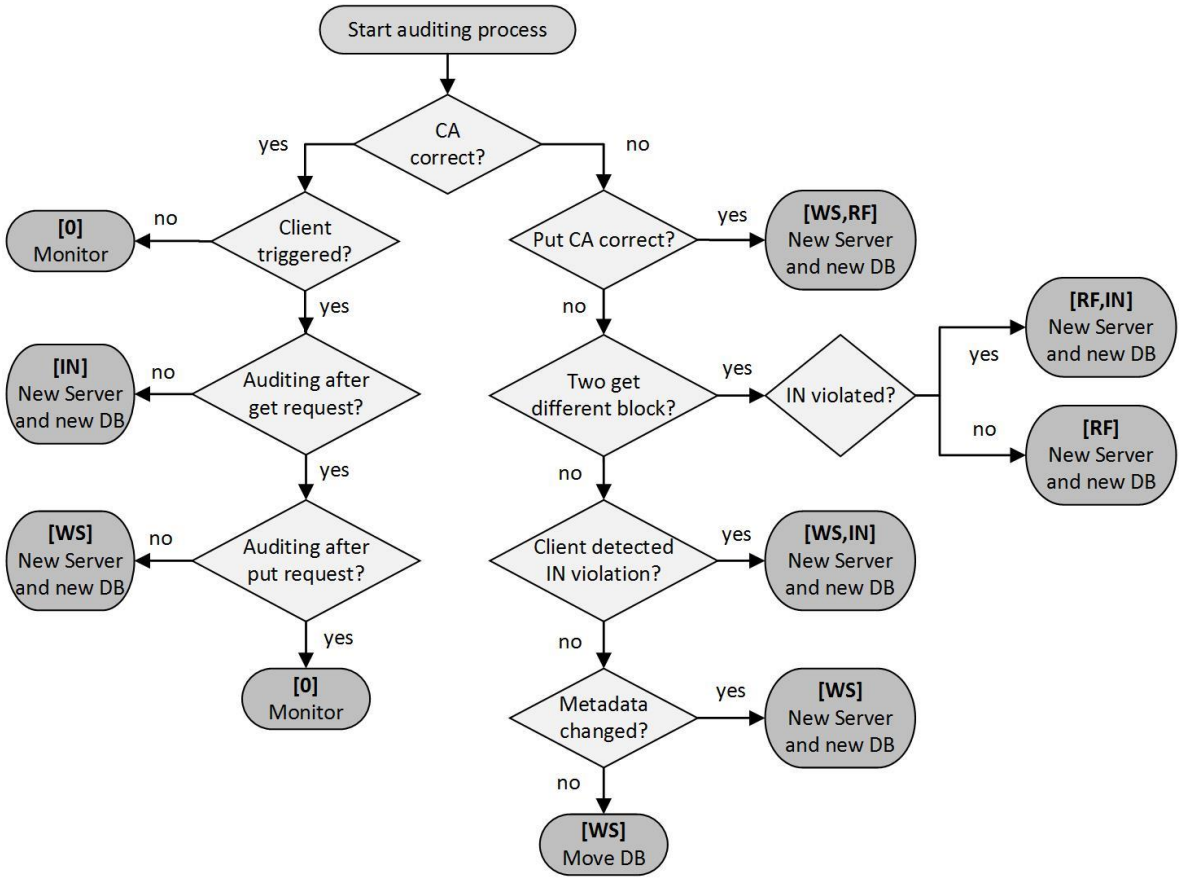

Figure 9. Auditing process

When the auditing process starts (either after an end of an epoch or after a notification from the Client), the Auditor first checks if the chain of attestations (denoted as CA) is correct (decision "*CA correct?*"). If the chain is correct, there are no violations that the Auditor is able to detect. There might have been an old content returned to the End-user at some point which would mean a violation of the fetch-modify consistency as defined in [10], but this cannot be detected by the Auditor.

If the chain of attestations is correct, the Auditor still has to consider possible system failures notified by the Client (decision "*Client triggered?*"). If the auditing process has not been triggered by the Client, then there is no violation of any SLA and the monitoring process can continue (result "*[0] Monitor.*"). If the auditing process has been triggered by the Client due to a detected integrity violation, then the result is a violation of IN or WS. As mentioned above, the Client itself can check correctness of the block hash of the data received with the `get` request and correctness of the chain hash for the data sent with the `put` request. The Auditor checks whether the Client triggered the process due to inconsistencies with the `get` (decision "*Auditing after get request?*") or the `put` request (decision "Auditing after *put request?*"). If the process was triggered due to one of these cases, a new Main Server and a new Main DB should be set up (results "*[IN] New Server and new DB.*" and "*[WS] New Server and new DB.*"). Otherwise there might have been an error with the Auditor, thus the monitoring can continue and no SLA violations have occurred (result "*[0] Monitor.*").

If the chain of attestations is not correct (the right side of the auditing process in Figure 9), the Auditor has to check for the occurrence of the fork attack (discussed in [10]) where the cloud maintains two copies of the data and conducting some writes and reads on the original data and others on its copy. In order to confirm or rule out the fork attack, the Auditor checks whether each `put` request has a correct chain of attestations behind it (decision "*Put CA correct?*").

If all put attestations have a correct chain behind them, then at some point the fork attack occurred which is a violation of WS. This is also a violation of RF since the End-user did not get the latest changes made by some other End-user. The Auditor can determine with which request the data has been forked. This information can later be used to set the system back to the time before the fork attack took place. In this case, a new Main Server and a new Main DB should be set up and the data should be restored from the backup to the version before the attack (result "*[WS,RF] New Server and new DB.*").

In the case where some `put` request does not have a correct chain of attestations behind it, the Auditor has to check whether there exist two `get` requests for the same block that received the same version number but different block content (decision "*Two get different block?*").

If such a pair of `get` requests exists, an attack might have happened. Two different End-users received different block data accompanied by the same version and block number (violation of RF). This is with high certainty due to a deliberate attack, thus the Main Server and the Main DB should be replaced. Note that also IN might be violated in both of these `get` requests (decision *"IN violated?"*). If IN is not violated in either, it means that either keys have been stolen or some old version of the block (with old hash) has been returned – this has to be checked and a special warning has to be sent to the Client if keys are stolen (result "*[RF] New Server and new DB.*"). If IN is violated in either of the two `get` requests, this might be due to a system error (like failure of a disk where database resides between the two requests). Regardless, it is better to go with a stricter remediation action replacing Main server and Main DB (result "*[RF,IN] New Server and new DB.*").

When such a pair of `get` requests (for the same block that received the same version but different block content) does not exist, this means that some `put` request was not executed

correctly and this represents a violation of the WS. In this case, the further process depends on whether the Client detected integrity violation (decision "*Client detected IN violation?*").

If the Client triggered the auditing process after detecting a violation of integrity (which the Auditor now confirms), both the Main Server and the Main DB have to be replaced (result "*[WS,IN] New Server and new DB.*").

On the contrary, if the Client did not detect any issues, the Auditor checks whether the block metadata (e.g., version number, block number) has been changed in a way that indicates a deliberate attack (decision "*Metadata changed?*").

If some element of the block metadata has been changed, for example, the returned chain hash is the one with the previous version number and previous block hash, this might indicate a rollback attack [11]. In this case, a new Main Server and a new Main DB have to be set up (result "*[WS] New Server and new DB.*").

On the contrary, if no metadata has been changed, then there are no indications for an attack. And since the assumption is that all issues are due to a system error, the Main Server should be switched to the Backup DB which would then take the role of the Main DB, and a new DB should be set up which would take the role of the Backup DB (result "*[WS] Move DB.*").

Whenever the Auditor detects or confirms a violation of any of the properties IN, WS, and RF, both CSP and affected End-users are notified about the violated property and the required remediation action. It is up to the Enforcement module to execute remediation actions and it is up to the affected End-users to claim compensations for the violation of the SLA. Of course, not all detected violations affect all End-users. For example, a violation of WS only affects End-users that have this property guaranteed in their SLA.

In the next section we focus on the monitoring data collected by the Monitoring Adapter. This information is not directly connected to WS, RF, and IN, but can indicate potential violations of these properties.

5.2.3. The Monitoring Adapter

In the SLA negotiation phase, End-users have the opportunity to choose which properties should be enforced and monitored by the CSP. For example, one End-user may want to have an assurance that IN, CO, WS, and RF will be sustained, whereas some other customer only wants a guarantee that WS will be respected. Each property may imply an additional service cost or it may have an effect on performance. Thus it is reasonable that some End-users may only want a certain feature to be granted and monitored.

Once the SLA is signed, and all components are deployed and configured, the End-user can start using the acquired service. In order to fulfil all commitments in the undertaken SLA, the CSP not only considers and manages notifications from the Auditor, but also uses the Monitoring Adapter, which oversees availability of both servers and both databases.

Monitoring Adapter continuously checks availability or responsiveness of both servers and databases. If at any moment any of them is unresponsive, the occurrence is notified to the Enforcement module since this may not only cause delays in the service but also errors that can affect IN, WS, or RF.

Moreover, if the End-user requests CO property, the Monitoring Adapter checks whether the Client component at the web store is of the correct version. Since the encryption/decryption is performed locally on the End-user's infrastructure, SPECS cannot monitor it and thus cannot guarantee that the Client correctly encrypts and decrypts data. Confidentiality (in terms of correctness of encryption/decryption) can be assured by providing the End-user with the latest and audited version of the Client component. SPECS is not responsible for any changes made to the Client code (by the End-user or by an attacker) after its installation.

In the next subsection we present remediation actions defined for (potential) SLA violations notified to the Enforcement module either by the Monitoring Adapter or the Auditor (for the initial version of remediation activities see D4.3.2).

5.2.4. Reaction to security incidents and system failures

As discussed in Section 5.2.3, the Monitoring Adapter continuously (before an End-user is provided with the link to the code) checks the Client version at the web store. If at some point for some reason the web store does not have the latest version of the Client components, SPECS uploads it (Monitoring Adapter notifies the Enforcement module, which then uploads the latest version of the Client to the web store). The process is depicted in Figure 10.

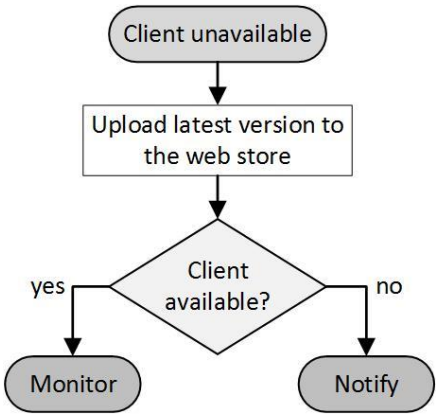

Figure 10. Uploading the new Client component

After the Enforcement module is notified about unavailability of any of the (application or database) servers, it first tries to restart it. If that solves the issue, monitoring continues. If restarting the unresponsive component does not help, the Enforcement module tries to deploy another instance of the unresponsive component. When there is a need to deploy a new database, the Enforcement also triggers backup (of data from the Main DB to the newly set up Backup DB) or restoration of data (from the Backup DB to the newly set up Main DB). If any of these steps fail to recover the system to a normal state, this may threaten the success of future `put` and `get` requests (i.e., validity of SLOs related to IN, WS, and RF metrics), thus the End-users should be notified about the event.

The remediation process in case of server failures (either for Main or Backup Server) is depicted in Figure 11, while the remediation process in case of database failures (either Main or Backup DB) is illustrated in Figure 12.

The Enforcement module also has to manage notifications of incidents and failures that are sent from the Auditor. As seen in Section 5.2.2, the Auditor not only detects violation of secure storage metrics, but also performs root cause analysis and identifies the optimal remediation

action. When a notification of a violation is sent to the Enforcement module, the Auditor reports about which metrics are violated (so that the CSP can determine the damage with respect to the affected SLAs), what the remediation plan is (to execute it), and which version of the data is the last correct one (to restore the data to the right version).

Remediation actions considered by the Auditor consist of either switching the Main DB to the Backup DB and setting up a new DB to take the role of the new Backup (case 1), or setting up a new pair of Main Server and Main DB components and restoring the data to a certain state (case 2).

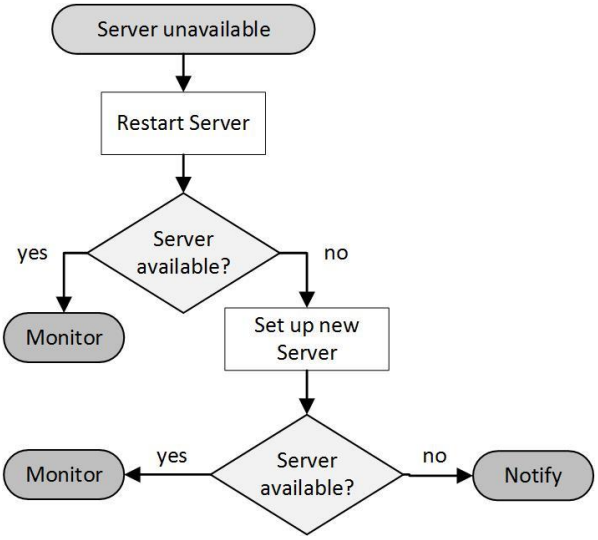

Figure 11. Remediation in case of (Main or Backup) Server failures

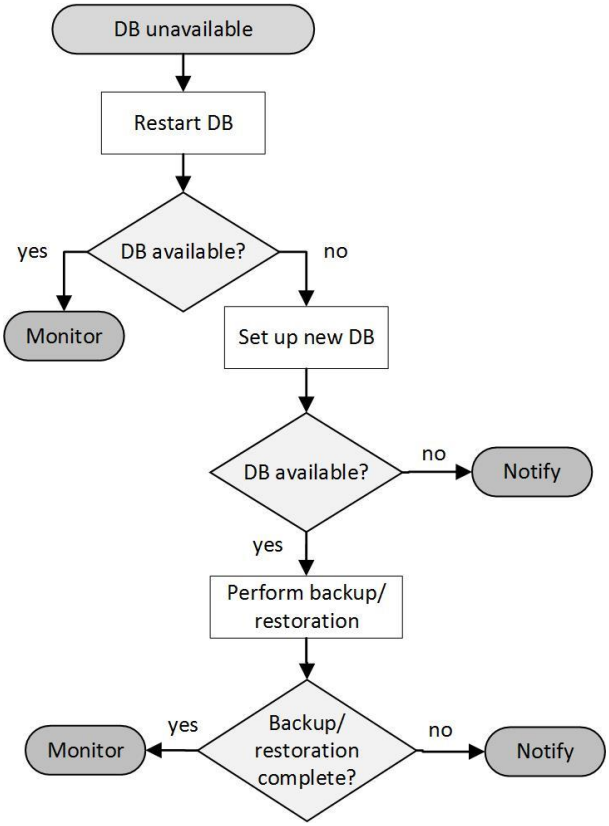

Figure 12. Remediation in case of (Main or Backup) DB failures

In the first case (see Figure 13), the Main Server is connected to the Backup DB which takes the role of the new Main DB. A new database is set up which takes the role of the Backup DB. Backup of the entire database is executed immediately.

When there is a need to set up a new Main Server and a new Main DB (case 2; see Figure 14), all data also have to be restored from the backup DB to a certain version as suggested by the Auditor.

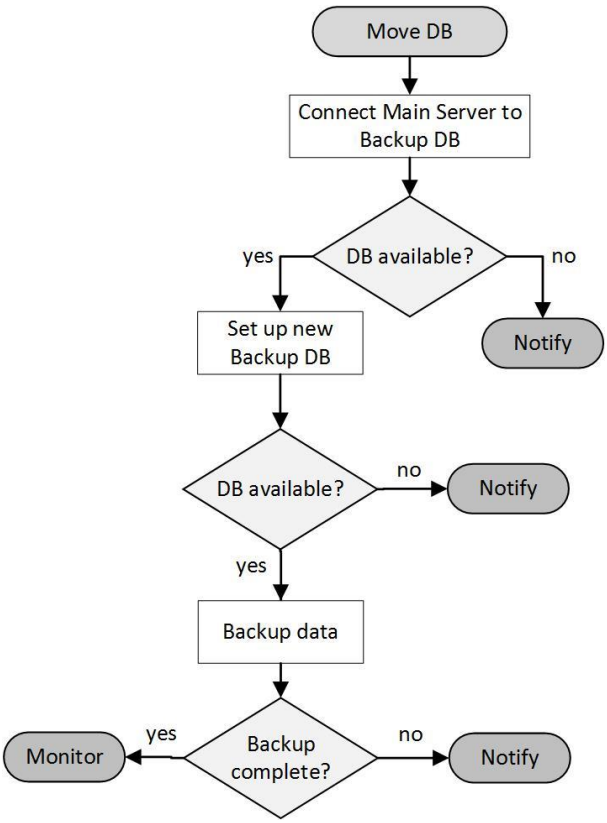

Figure 13. Moving Main DB

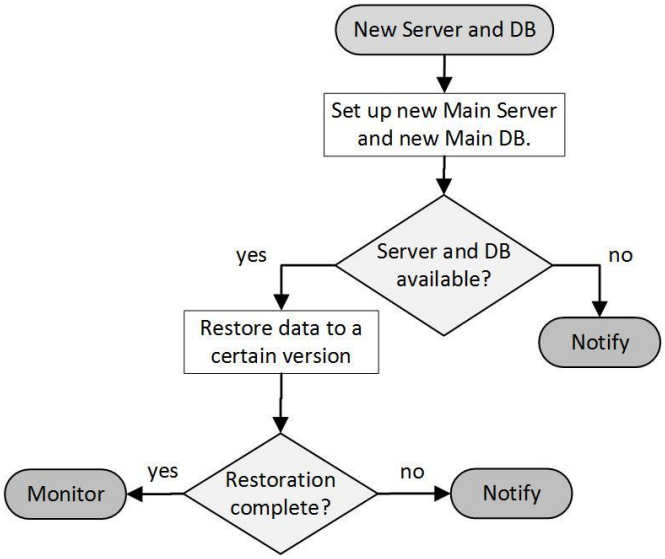

Figure 14. Setting up new Main Servers

More details about how the Enforcement module automatically executes introduced remediation actions are provided in deliverable D4.3.3. Technical details about the implementation of the DBB and E2EE security mechanisms are in deliverable D5.2.2.

6. Conclusions

In this deliverable we presented the development process of the Secure Storage application and discussed its characteristics:

- We summarized the user story and validation scenarios that define the service and the functionalities that the developed application offers;
- We described the artifacts that needed to be prepared for the application deployment;
- We illustrated the architecture of the application by presenting the role of the SPECS framework, by describing the role of the cloud storage provider Koofr, and by providing a comprehensive description of functionalities of both security mechanisms integrated with the application.

The described application facilitates some novel approaches to offering security assurances in the cloud storage domain. Apart from the automated provisioning, negotiating, enforcing, and monitoring cloud services through SLAs, the true value of the application lies in the ability to automatically enforce, monitor, and react to violations of confidentiality, integrity, write-serializability, and read-freshness. The idea to include these properties in SLAs is not new (see the CloudProof system introduced in [3]). However, the authors of the CloudProof only detect violations of some security properties, but do not try to determine the root cause. We go one step further and provide the extension of the mechanism that is able to distinguish among different types of attacks in case of their violations. Root cause analysis is important not only because it gives an insight into what is going on in the system, but also because with that kind of additional information CSPs can choose and apply optimal reactive measures to recover from the incident. On top of the extension of the CloudProof system, we provide an improvement associated to the implementation of the encryption/decryption process, which will be further elaborated in deliverable D5.2.2 (where we also discuss lessons learnt in T5.2).

The following table presents objectives associated to this task and reports the outcomes which verify the benefits of the results achieved in this task.

Objective	Result
• SO5.2: Design and implement real applications using the SPECS platform	• The developed application integrates the SPECS framework to enable the automated management of cloud storage SLAs. It facilitates a novel approach to monitoring, notifying, and reacting to security incidents. • The developed application integrates two SPECS developed security mechanisms which can enhance security level of any cloud storage solution. Moreover, the mechanisms form a so called proof-based system which not only detects SLA violations but provides their proofs. • The developed cloud storage security mechanisms improve the current state of the art with (i) the ability to identify the root cause of violations of integrity, write-serializability, and read-freshness, and (ii) the ability to automatically remediate them [1]. • The developed client-side encryption mechanism improves the current state of the art with respect to the speed of encryption/decryption by implementing corrections in certain crypto libraries (see D5.2.2). • The developed application is part of the SPECS solution portfolio [2].

Table 3. Objectives and results of task T5.2

After the end of the project, the plan is to integrate a part of the application (the E2EE and DBB mechanisms and modules that enable the automated remediation) with Koofr infrastructure to offer to its users not only better but also more transparent security assurances. In practice, Koofr will integrate DBB and E2EE mechanisms into their own infrastructure. The Servers developed in SPECS will be used to store and backup data uploaded by Koofr users, the Auditor will be used for advanced monitoring of the stored data, and the Client will be used to offer Koofr users the client-side encryption. These artifacts will enable Koofr to offer to its users end-to-end encryption and advanced monitoring for potential security incidents and system failures.

DBB and E2EE mechanisms have been developed so that they can be used as standalone mechanisms for providing advanced security features, they can be used together with the complete Secure Storage application supporting automated implementation and remediation, or they can be used in other contexts (for example, see AAAaaS application introduced in D5.4), thus ensuring strong appeal with potential users. For more details about exploitability and marketability of the developed mechanisms, see D6.2.3.

The last deliverable of the task T5.2 (namely D5.2.2) presents all technical details associated to the development of the Secure Storage application. In particular, deliverable D5.2.2 presents the implementation details of the security mechanisms integrated into the application and provides installation and usage guides along with the report about the testing activities associated to the developed application.

7. Bibliography

[1] M. Stopar, J. Modic, D. Petcu, M. Rak, "*Towards a Proof-Based SLA Management Framework – The SPECS Approach*", CLOSER 2016.

[2] SPECS, "*End-2-End Encryption*", 2015. Available online, http://www.specs-project.eu/solutions-portofolio/e2ee/, last accessed in February 2016.

[3] R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, L. Zhuang, "*Enabling Security in cloud Storage SLAs with CloudProof*", in Proceedings of the USENIX ATC'11, the USENIX Annual Technical Conference, pp. 31, 2011.

[4] SPECS, "*SPECS Web Container App Description*", 2014. Available online, https://bitbucket.org/specs-team/specs-app-webcontainer, last accessed in February 2016.

[5] SPECS, "*SPECS Chef Repository – Enforcement*", 2015. Available online, https://bitbucket.org/specs-team/specs-core-enforcement-repository/overview, last accessed in February 2016.

[6] Chef Software, "*Chef*", 2008. Available online, https://www.chef.io/, last accessed in February 2016.

[7] Koofr d.o.o., "*Koofr*", 2015. Available online, http://koofr.eu/, last accessed in February 2016.

[8] NIST National Institute of Standards and Technology, "*Security and privacy controls for federal information systems and organizations*", NIST 800-53v4, 2013. Available online, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf, last accessed in February 2016.

[9] Cloud Security Alliance, "*Cloud Controls Matrix Working Group*", 2015. Available online, https://cloudsecurityalliance.org/group/cloud-controls-matrix/, last accessed in February 2016.

[10] J. Li, M. Krohn, D. Mazierès, D. Shasha, "*Secure untrusted data repository (SUNDR)*", 2004. In Proceedings of the OSDI'04, the 6th Conference on Symposium on Operating Systems Design & Implementation, USENIX, pp. 9, 2004.

[11] J. Feng, Y. Chen, D. Summerville, W. S. Ku, Z. Su, "*Enhancing cloud storage security against roll-back attacks with a new fair multiparty non-repudiation protocol*", 2011. In Proceedings of the CCNC'11, the IEEE Consumer Communications and Networking Conference, pp 521–522, 2011.

Appendix 1. The Secure Storage SLA Template

In the following we report the SLA Template (in XML format) created for the Secure Storage application (with default values for all associated security metrics). We highlight the important parts of it in yellow.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wsag:AgreementOffer
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:specs="http://www.specs-project.eu/resources/schemas/xml/SLAtemplate"
    xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:nist="http://www.specs-
                project.eu/resources/schemas/xml/control_frameworks/nist"
    xmlns:ccm="http://www.specs-project.eu/resources/schemas/xml/control_frameworks/ccm">


    <wsag:Name>SECURE_STORAGE</wsag:Name>


    <wsag:Context>
        <wsag:AgreementInitiator>specs-customer-2</wsag:AgreementInitiator>
        <wsag:AgreementResponder>$SPECS-APPLICATION</wsag:AgreementResponder>
        <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
        <wsag:ExpirationTime>2016-04-30T06:00:00+03:00</wsag:ExpirationTime>
        <wsag:TemplateName>SECURE_STORAGE</wsag:TemplateName>
    </wsag:Context>


    <wsag:Terms>
        <wsag:All>
            <wsag:ServiceDescriptionTerm
                wsag:Name="Secure Storage"
                wsag:ServiceName="SecureStorage">
                <specs:serviceDescription>
                    <specs:serviceResources>
                        <specs:resourcesProvider
                            id="aws-ec2"
                            name="Amazon"
                            zone="us-east-1"
                            maxAllowedVMs="20"
                            description=""
                            label="">
                            <specs:VM
                                appliance="us-east-1/ami-ff0e0696"
                                hardware="t1.micro"
                                description="open suse 13.1 on amazon EC2"/>
                        </specs:resourcesProvider>
                    </specs:serviceResources>

                    <specs:capabilities>
                        <specs:capability
                           id="DBB"
                           name="Database and Backup as-a-service"
                           description="Capability of surviving to incidents that
                                        compromise the availability and/or integrity of
                                        data stored remotely by providing backup
                                        service and the detection of WS and RF
                                        violations"
                           mandatory="false">
                            <specs:controlFramework
                                id="CCM_v3.0"
                                frameworkName="CCM Control framework v3.0">

                                <specs:CCMsecurityControl
                                    id="IVS-02"
```

```
                        name="Infrastructure and Virtualization Security -
                             Change Detection"
                        control_domain="IVS">
                        <ccm:description>The provider shall ensure the
                        integrity of all virtual machine images at all times.
                        Any changes made to virtual machine images must be
                        logged and an alert raised regardless of their
                        running state (e.g. dormant, off, or running).
                        </ccm:description>
                        <ccm:importance_weight>MEDIUM</ccm:importance_weight>
                    </specs:CCMsecurityControl>

                    <specs:CCMsecurityControl
                        id="BCR-01"
                        name="Business Continuity Mgmt and Op Resilience -
                              Business Continuity Planning"
                        control_domain="BCR">
                        <ccm:description>A consistent unified framework for
                        business continuity planning and plan development
                        shall be established, documented and adopted to
                        ensure all business continuity plans are consistent
                        in addressing priorities for testing, maintenance,
                        and information security requirements.
                        </ccm:description>

                        <ccm:importance_weight>MEDIUM</ccm:importance_weight>
                    </specs:CCMsecurityControl>

                    <specs:CCMsecurityControl
                        id="BCR-11"
                        name="Business Continuity Mgmt and Op Resilience -
                              Retention Policy" control_domain="BCR">
                        <ccm:description>Policies and procedures shall be
                        established, for defining and adhering to the
                        retention period of any critical asset as per
                        established policies and procedures, as well as
                        applicable legal, statutory, or regulatory compliance
                        obligations. Backup and recovery measures shall be
                        incorporated as part of business continuity planning
                        and tested accordingly for effectiveness.
                        </ccm:description>
                        <ccm:importance_weight>MEDIUM</ccm:importance_weight>
                    </specs:CCMsecurityControl>

                    <specs:CCMsecurityControl
                        id="AIS-03"
                        name="Application and Interface Security - Data
                              Integrity" control_domain="AIS">
                        <ccm:description>Data input and output integrity
                        routines (i.e., reconciliation and edit checks) shall
                        be implemented for application interfaces and
                        databases to prevent manual or systematic processing
                        errors, corruption of data, or misuse.
                        </ccm:description>
                        <ccm:importance_weight>MEDIUM</ccm:importance_weight>
                    </specs:CCMsecurityControl>
                </specs:controlFramework>
            </specs:capability>

            <specs:capability
                id="E2EE"
                name="End-to-end Encryption"
                description="Capability of providing client-side encryption
                             enforcing confidentiality."
                mandatory="false">
                <specs:controlFramework
                    id="CCM_v3.0"
```

```
                        frameworkName="CCM Control framework v3.0">
                        <specs:CCMsecurityControl
                            id="EKM-01"
                            name="Encryption and Key Management - Entitlement"
                            control_domain="EKM">
                            <ccm:description>Keys must have identifiable owners
                            (binding keys to identities) and there shall be key
                            management policies.
                            </ccm:description>
                            <ccm:importance_weight>MEDIUM</ccm:importance_weight>
                        </specs:CCMsecurityControl>

                        <specs:CCMsecurityControl
                            id="EKM-03"
                            name="Encryption and Key Management - Sensitive Data
                                  Protection" control_domain="EKM">
                            <ccm:description>Policies and procedures shall be
                            established, for the use of encryption protocols for
                            protection of sensitive data in storage (e.g., file
                            servers, databases, and end-user workstations), data
                            in use (memory), and data in transmission (e.g.,
                            system interfaces, over public networks, and
                            electronic messaging).
                            </ccm:description>
                            <ccm:importance_weight>MEDIUM</ccm:importance_weight>
                        </specs:CCMsecurityControl>
                    </specs:controlFramework>
                </specs:capability>
            </specs:capabilities>

            <specs:security_metrics>
                <specs:Metric
                    name="Write-Serializability"
                    referenceId="write_serializability_M17">
                    <specs:MetricDefinition>
                        <specs:unit name="">
                            <specs:enumUnit>
                               <specs:enumItemsType>boolean</specs:enumItemsType>
                                <specs:enumItems>
                                    <specs:enumItem></specs:enumItem>
                                    <specs:enumItem></specs:enumItem>
                                </specs:enumItems>
                            </specs:enumUnit>
                        </specs:unit>
                        <specs:scale>
                            <specs:Quantitative>Ratio</specs:Quantitative>
                        </specs:scale>
                        <specs:expression>The activation of write
                        serializability</specs:expression>
                        <specs:definition>This metric ensures the EU consistency
                        among updates of the stored data. In case of WS
                        violations, the EU will be notified and the system will
                        be restored to the state of the last completed
                        update.</specs:definition>
                        <specs:note></specs:note>
                    </specs:MetricDefinition>
                </specs:Metric>

                <specs:Metric
                    name="Read-Freshness"
                    referenceId="read_freshness_M18">
                    <specs:MetricDefinition>
                        <specs:unit name="">
                            <specs:enumUnit>
                               <specs:enumItemsType>boolean</specs:enumItemsType>
                                <specs:enumItems>
                                    <specs:enumItem></specs:enumItem>
```

```
                                    <specs:enumItem></specs:enumItem>
                                </specs:enumItems>
                            </specs:enumUnit>
                        </specs:unit>
                        <specs:scale>
                            <specs:Quantitative>Ratio</specs:Quantitative>
                        </specs:scale>
                        <specs:expression>The activation of read
                        freshness</specs:expression>
                        <specs:definition>This metric ensures the EU that the
                        requested data will always be fresh as of the last
                        update. In case of RF violations, the EU will be notified
                        and the system will be restored to the state of the last
                        completed backup.</specs:definition>
                        <specs:note></specs:note>
                    </specs:MetricDefinition>
                </specs:Metric>

                <specs:Metric
                    name="Integrity"
                    referenceId="integrity_M25">
                    <specs:MetricDefinition>
                        <specs:unit name="">
                            <specs:enumUnit>
                               <specs:enumItemsType>boolean</specs:enumItemsType>
                                <specs:enumItems>
                                    <specs:enumItem></specs:enumItem>
                                    <specs:enumItem></specs:enumItem>
                                </specs:enumItems>
                            </specs:enumUnit>
                        </specs:unit>
                        <specs:scale>
                            <specs:Quantitative>Ratio</specs:Quantitative>
                        </specs:scale>
                        <specs:expression>The activation of
                        integrity.</specs:expression>
                        <specs:definition>This metric ensures the EU
                        integrity of the stored data.</specs:definition>
                        <specs:note></specs:note>
                    </specs:MetricDefinition>
                </specs:Metric>

                <specs:Metric
                    name="Confidentiality"
                    referenceId="confidentiality_M19">
                    <specs:MetricDefinition>
                        <specs:unit name="">
                            <specs:enumUnit>
                               <specs:enumItemsType>boolean</specs:enumItemsType>
                                <specs:enumItems>
                                    <specs:enumItem></specs:enumItem>
                                    <specs:enumItem></specs:enumItem>
                                </specs:enumItems>
                            </specs:enumUnit>
                        </specs:unit>
                        <specs:scale>
                            <specs:Quantitative>Ratio</specs:Quantitative>
                        </specs:scale>
                        <specs:expression>The existence of
                        certification.</specs:expression>
                        <specs:definition>This metric ensures the EU
                        confidentiality of the stored data. Confidentiality is
                        enforced with end-2-end encryption provided by the Client
                        component. We guarantee that the Client component is
                        audited and thus (used as is) grants security of
                        encryption.</specs:definition>
                        <specs:note></specs:note>
```

```
                    </specs:MetricDefinition>
                </specs:Metric>
            </specs:security_metrics>
        </specs:serviceDescription>
    </wsag:ServiceDescriptionTerm>

    <wsag:ServiceProperties
        wsag:Name="//specs:capability[@id='DBB']"
        wsag:ServiceName="SecureStorage">
        <wsag:VariableSet>
            <wsag:Variable
                wsag:Name="specs_DBB_M17"
                wsag:Metric="write_serializability_M17">
                <wsag:Location>
                    //specs:CCMsecurityControl[@id='IVS-02'] |
                    //specs:CCMsecurityControl[@id='BCR_01'] |
                    //specs:CCMsecurityControl[@id='BCR_11']
                </wsag:Location>
            </wsag:Variable>

            <wsag:Variable
                wsag:Name="specs_DBB_M18"
                wsag:Metric="read_freshness_M18">
               <wsag:Location>
                   //specs:CCMsecurityControl[@id='AIS-03'] |
                   //specs:CCMsecurityControl[@id='BCR_01'] |
                   //specs:CCMsecurityControl[@id='BCR_11']
               </wsag:Location>
            </wsag:Variable>

            <wsag:Variable
                wsag:Name="specs_DBB_M25"
                wsag:Metric="integrity_M25">
               <wsag:Location>
                   //specs:CCMsecurityControl[@id='AIS-03']
               </wsag:Location>
            </wsag:Variable>
        </wsag:VariableSet>
    </wsag:ServiceProperties>

    <wsag:ServiceProperties
        wsag:Name="//specs:capability[@id='E2EE']"
        wsag:ServiceName="SecureStorage">
        <wsag:VariableSet>
            <wsag:Variable
                wsag:Name="specs_e2ee_M19"
                wsag:Metric="confidentiality_M19">
                <wsag:Location>
                    //specs:CCMsecurityControl[@ccm:id='EKM-01'] |
                    //specs:CCMsecurityControl[@ccm:id='EKM-03']
                </wsag:Location>
            </wsag:Variable>
        </wsag:VariableSet>
    </wsag:ServiceProperties>

    <wsag:GuaranteeTerm
        wsag:Name="//specs:capability[@id='DBB']"
        wsag:Obligated="ServiceProvider">
        <wsag:ServiceScope wsag:ServiceName="SecureStorage"/>
        <wsag:QualifyingCondition>false</wsag:QualifyingCondition>
        <wsag:ServiceLevelObjective>
            <wsag:CustomServiceLevel>
                <specs:objectiveList>
                    <specs:SLO SLO_ID="DBB_slo1">
                        <specs:MetricREF>specs_DBB_M17</specs:MetricREF>
                        <specs:SLOexpression>
                            <specs:oneOpExpression>
```

```
                                    <specs:operator>eq</specs:operator>
                                    <specs:operand>yes</specs:operand>
                                </specs:oneOpExpression>
                            </specs:SLOexpression>
                            <specs:importance_weight>MEDIUM</specs:importance_weight>
                        </specs:SLO>

                        <specs:SLO SLO_ID="DBB_slo2">
                            <specs:MetricREF>specs_DBB_M18</specs:MetricREF>
                            <specs:SLOexpression>
                                <specs:oneOpExpression>
                                    <specs:operator>eq</specs:operator>
                                    <specs:operand>yes</specs:operand>
                                </specs:oneOpExpression>
                            </specs:SLOexpression>
                            <specs:importance_weight>MEDIUM</specs:importance_weight>
                        </specs:SLO>

                        <specs:SLO SLO_ID="DBB_slo3">
                            <specs:MetricREF>specs_DBB_M25</specs:MetricREF>
                            <specs:SLOexpression>
                                <specs:oneOpExpression>
                                    <specs:operator>eq</specs:operator>
                                    <specs:operand>yes</specs:operand>
                                </specs:oneOpExpression>
                            </specs:SLOexpression>
                            <specs:importance_weight>MEDIUM</specs:importance_weight>
                        </specs:SLO>
                    </specs:objectiveList>
                </wsag:CustomServiceLevel>
            </wsag:ServiceLevelObjective>
            <wsag:BusinessValueList></wsag:BusinessValueList>
        </wsag:GuaranteeTerm>

        <wsag:GuaranteeTerm
            wsag:Name="//specs:capability[@id='E2EE']"
            wsag:Obligated="ServiceProvider">
            <wsag:ServiceScope wsag:ServiceName="SecureStorage"/>
            <wsag:QualifyingCondition>false</wsag:QualifyingCondition>
            <wsag:ServiceLevelObjective>
                <wsag:CustomServiceLevel>
                    <specs:objectiveList>
                        <specs:SLO SLO_ID="e2ee_slo1">
                            <specs:MetricREF>specs_e2ee_M19</specs:MetricREF>
                            <specs:SLOexpression>
                                <specs:oneOpExpression>
                                    <specs:operator>eq</specs:operator>
                                    <specs:operand>yes</specs:operand>
                                </specs:oneOpExpression>
                            </specs:SLOexpression>
                            <specs:importance_weight>MEDIUM</specs:importance_weight>
                        </specs:SLO>
                    </specs:objectiveList>
                </wsag:CustomServiceLevel>
            </wsag:ServiceLevelObjective>
            <wsag:BusinessValueList></wsag:BusinessValueList>
        </wsag:GuaranteeTerm>
    </wsag:All>
  </wsag:Terms>
</wsag:AgreementOffer>
```