MEDINA

Deliverable D4.1

Tools and techniques for the tanagement and evaluation of cloud security certifications – v1

Editor(s):	Immanuel Kunz
Responsible Partner:	Fraunhofer Institute for Applied and Integrated Security (AISEC)
Status-Version:	Final – v1.1
Date:	30.09.2022
Distribution level (CO, PU):	PU

Project Number:	952633
Project Title:	MEDINA

Title of Deliverable:	Deliverable D4.1 – Tools and Techniques for the Management and Evaluation of Cloud Security Certifications
Due Date of Delivery to the EC	31.10.2021

Workpackage responsible for the Deliverable:	WP4 – Continuous Life-Cycle Management of Cloud Security Certifications
Editor(s):	Immanuel Kunz (FhG)
Contributor(s):	AISEC, XLAB, TECNALIA, NIXU, Bosch
Reviewer(s):	Björn Fanta (Fabasoft), Cristina Martinez (TECNALIA)
Approved by:	All Partners
Recommended/mandatory readers:	WP3, WP5, WP6

Abstract:	This deliverable contains contributions towards the automation of certification evaluation and management steps, as well as risk assessments and possible mitigations regarding the protection of evidence and certificate management. It is the first WP4 deliverable and will be followed by two further iterations (D4.2 and D4.3). It is the result of work on the tasks T4.1, T4.2, and T4.4.
Keyword List:	Certificate Evaluation, Certificate Management, Distributed Ledger Technologies, Smart Contracts
Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/
Disclaimer	This document reflects only the author's views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein.

Document Description

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	08.12.2020	Initial TOC	FhG
v0.2	23.09.2021	First draft version	FhG, XLAB, TECNALIA
v0.3	07.10.2021	Improved second draft	FhG, XLAB, TECNALIA
v0.4	15.10.2021	Pre-final version with integrated feedback	FhG, XLAB, TECNALIA
v0.5	28.10.2021	Final version with integrated feedback from the internal reviewer	FhG, XLAB, TECNALIA
v1.0	28.10.2021	Ready for submission	TECNALIA
v1.01	03.08.2022	Comments from EU review implemented. Ready for internal review	FhG
v1.02	22.08.2022	Addressed all comments received in the internal QA review	FhG, Fabasoft
v1.1	30.09.2022	Ready for submission	TECNALIA

Table of Contents

List of Tables

List of Figures

Terms and abbreviations

API	Application Programming Interface
BFT	Byzantine Fault Tolerance
CAB	Conformity Assessment Body
CCE	Continuous Certification Evaluation
CIA	Confidentiality, Integrity, and Availability
CFT	Crash Fault Tolerance
CSC	Cloud Service Customer
CSP	Cloud Service Provider
DID	Decentralized Identifiers
DLT	Distributed Ledger Technologies
DoS	Denial of Service
EBSI	European Blockchain Services Infrastructure
EC	European Commission
GA	Grant Agreement to the project
GPL	General Public License
gRPC	Google Remote Procedure Call
HTTP	Hypertext Transfer Protocol
IPFS	Interplanetary File System
KPI	Key Performance Indicator
HW	Hardware
IaaS	Infrastructure as a Service
IBFT	Istanbul Byzantine Fault Tolerance
IoT	Internet of Things
LCM	Life-Cycle Manager
LGPL	Lesser General Public License
MitM	Man in the Middle
MIP	Moving Intervals Process
PaaS	Platform as a Service
PBFT	Practical Byzantine Fault Tolerance
PoET	Proof of Elapsed Time
PoA	Proof of Authority
PoS	Proof-of-Stake
PoW	Proof-of-Work
QHP	Quantitative Hierarchy Process
QPT	Quantitative Policy Trees
SaaS	Software as a Service
SLA	Service Level Agreement
SLO	Service Level Objective
secSLA	Secure Service Level Agreement
SPoF	Single Point of Failure
SSI	Self-Sovereign Identities
SW	Software
TOC	Target Of Certification
TOM	Technical and Organizational Measure
TPS	Transactions Per Second

UI	User Interface
WP	Work Package

Executive Summary

Evaluating, managing, and protecting certificates and their underlying evidence are challenging tasks, which do, however, have potential for automation. This deliverable targets these challenges in three parts.

First, an approach and technical prototype for the evaluation of assessment results is presented (Section 4). Assessment results are created in Work Package 3 and are forwarded to this evaluation component. Here, they are aggregated in a continuous manner, building a tree structure of the certification that is both machine-readable and quickly understandable by humans, e.g., auditors.

Second, a risk analysis is conducted for the evidence and assessment results that underly every certificate's state changes. Also, possible mitigations are evaluated, including blockchain and many blockchain-like technologies that have emerged recently (Section 4.3).

Third, an approach and initial prototype for a certificate life-cycle manager is presented (Section 6). It implements a state machine that reflects the EUCS [1] defined certificate states. Its approach includes an evaluation of different technological approaches to implementing and securing certificates, including smart contracts and self-sovereign identities.

Note also that we describe background concepts and related work in Section 2, and present the overall architecture of the WP4 components in Section 3. This deliverable is tightly connected to D3.1 [2]and D3.4 [3], since it directly processes the results of the components developed in WP3. The overall purpose and integration of the components described here in the MEDINA framework is furthermore described in WP5 [4].

This deliverable presents the first version of the WP4 components, except of the Risk Assessment component – work on the Risk Assessment component starts after the submission of this deliverable and will be described in the deliverables D4.4 [5] and D4.5 [6]. Two more versions of the deliverable at hand (D4.2, D4.3) will be created in the future. The following additions are planned for these future iterations:

- Section 2 will be extended with description of more related literature.
- The approach to evaluating certifications based on assessment results will be extended, e.g., to calculate KPIs, like the average time-to-fix of non-compliances. Also, the prototypes will be integrated with the other components.
- The evaluation of blockchain-like technologies in Section 4.3 will be extended.
- The functionalities required for the digital audit trail process will be studied and identified to be included in the MEDINA trustworthy management system, improving its usability for auditors.
- Section 6, i.e., the life-cycle manager, will be improved with an extended evaluation of smart contracts, and an enhanced prototype implementation.
- The use of SSI for verifiable digital conformity attestation certificates will be improved by means of a deeper analysis of the way to apply SSI and will be integrated with the prototype implementation.
- Finally, the risk assessment component will be developed and integrated, which will be described in a separate deliverable.

Overall, the results described in this deliverable present a first iteration towards the automation of the management of certificates. At various points, it identifies risks, possible mitigations, and evaluation considerations that may lead different users to different results. As such, it guides

CSPs towards appropriate solutions and provides prototype implementations to simplify their adoption.

1 Introduction

Enabling the continuous certification of cloud services requires a number of innovations. Apart from a common abstraction and language for requirements and metrics (see D2.1 [7]), and a continuous gathering and assessment of evidence (see D3.1 [2]), the final aggregation and evaluation of the assessment presents a major challenge.

Normally, manual audits by third party auditors are conducted to verify a set of pre-defined requirements at a point in time. Consequently, the decision of issuing a certificate is made by humans considering various sources of evidence, like documentation, data samples, and interviews. Auditors will therefore not only evaluate specific pieces of evidence, but also the gathered evidence as a whole. This process allows for some amount of consideration by the auditors who can evaluate the fulfilment of requirements depending on the context.

The technical implementation of this process has advantages and disadvantages: On the one hand, an automated certification process can provide continuous auditing, improved traceability of decisions, and a more standardized process. On the other hand, an automated certification process has a rigid focus on evidence that can be gathered technically, e.g., configurations of cloud resources. Consequently, where human auditors may weigh different kinds of evidence considering the service and its context, an automated implementation needs clearly defined requirements and decision criteria independently of the service's context.

Furthermore, certificates need to be managed. In the traditional certification process, where audits are not done continuously, a certificate can simply be published, for instance in a public registry. In the continuous model, however, any newly gathered evidence can have a major impact on the certificate's state. Its state therefore needs to be evaluated continuously as well, and its publication needs to be managed – possibly automatically with implications for the CSP's reputation.

Finally, the results generated by such components, as well as the components' logic, also need to be protected against intentional and unintentional threats. For instance, certificate state changes need to be verifiable to establish trust into this automated process.

1.1 About this Deliverable

The main goal of this deliverable is to describe MEDINA's contribution towards the continuous evaluation of security assessments of cloud services. These contributions include an approach for continuously aggregating assessment results, as well as deriving a decision about the certificate state.

The proposed approach for continuous aggregation of assessment results represents a certification as a tree-like structure that is evaluated based on its leaves—the assessment results. Thereafter, the development of a risk assessment component is foreseen that processes the results qualitatively to consider service-specific criteria, like especially impactful resources. Finally, the life-cycle management component reflects a certificate's state as a traceable state machine. It makes state change decisions based on deviation reports by the risk assessment. To protect its correct execution, different technologies, like smart contracts, are considered and evaluated.

The deliverable furthermore includes the research results for the protection of evidence and assessment results, whose implementation is described in D3.1 [2]. This includes the identification of risks for these artefacts, and an evaluation of possible mitigative technologies.

These contributions aim at yielding the advantages mentioned above, e.g., an improved traceability and automation, while at the same time addressing potential risks and challenges of managing and protecting certificates.

1.2 Document Structure

The deliverable is structured as follows. Chapter 2 briefly describes some background concepts and related work for the following chapters. In Chapter 3, the overall architecture and goals of the developed components are described. Thereafter, Chapter 4 presents the MEDINA approach to evaluate assessment results, aggregating them into a certification tree. Next, the risks that evidences and assessment results are exposed to are analysed in Chapter 5, laying the groundwork for the implementation of a trustworthiness system. Chapter 6 then presents the approach and current status of the life-cycle management component. Finally, Chapter 7 concludes the deliverable.

2 Background and Related Work

This section explains background concepts and some related literature that build the foundation for the work described in the rest of the deliverable.

2.1 Evaluating Cloud Security Certifications

Evaluation of security compliance in MEDINA starts with the gathering of evidence in WP3 components (see [2] [3]). Security assessment components assess these evidence based on the target values as configured for the specific requirement and provide their output (assessment results with the state of fulfilment of a specific metric for a specific monitored resource) to the Continuous Certification Evaluation component. If the assessment result value represents the lowest-level information about the certification state, the role of the Continuous Certification Evaluation component is to combine the received assessment results into information about the fulfilment of higher-level certification objects: requirements, controls, control groups, and the selected certificate scheme in its entirety. This information does not directly determine the cloud service's eligibility for a certificate, but serves as input for other components, the Risk Assessment and Optimisation Framework (which will be described in the deliverables D4.4 [5] and D4.5 [6]) and the Automated Certificate Lifecycle Management (see Sections 2.3 and 6), as well as for easy visualisation of the certificate state for the users (CSPs and auditors).

To assist the design of the Continuous Certification Evaluation component, previous research about similar problems was consulted. Luna et al. [8] presented two methods (based on Quantitative Policy Trees (QPT) [9] and Quantitative Hierarchy Process (QHP) [10]) for quantitatively assessing whether (and to what extent) a CSP fulfils the security requirements expressed by a customer, and the general level of security offered by a CSP. Their method is based on cloud security Service Level Agreements (secSLAs), which consist of various Service Level Objectives (SLOs) that map to one or more measurable metrics. Cloud Service Customers (CSCs) express their security needs by defining thresholds for the values of metrics and weights (importance) of the individual SLOs (QPT) or all levels of the SLA hierarchy (QHP). The CSPs' secSLAs are evaluated with respect to the customer's security requirements to output a ranking of CSPs according to their level of fulfilment of these requirements.

Modic et al. [11] improved the computational efficiency of the previously presented QHP method and developed a high-performance technique, Moving Intervals Process (MIP), which, beside checking the fulfilment and potential under-provisioning of CSC's requirements, also rates CSPs based on how much the customer's requests can be over-provisioned by the cloud service. Like QHP, MIP also uses calculations based on weighted arithmetic mean to aggregate values of SLOs to all levels of the secSLA hierarchy. According to the level of fulfilment of a customer's requirement, MIP assigns values to SLOs on the interval [0,2] where values less than 1 represent under-provisioning, 1 is assigned where the CSP exactly meets the requirement, and values greater than 1 represent over-provisioning. Because some of the values on the same hierarchy level can be greater than 1 and others less, the aggregated value can result in apparent over-provisioning (>1) even though some child values do not even meet the customer's requirement. The authors suggested a correction to the scores to eliminate this masquerading effect. Both of the mentioned methodologies for cloud security evaluation were (developed and) used in the EU FP7 project SPECS [12].

Maroc and Zhang [13] proposed a cloud security evaluation approach that additionally features a risk-driven selection of evaluation criteria and considers multiple factors in weighting of criteria: user's preferences, criteria interdependencies, the type of cloud service (IaaS / PaaS /

SaaS) that determines the user's level of control, as well as relations between threats and vulnerabilities, their risk (likelihood and impact), and security controls.

In MEDINA, the Continuous Evaluation component does not give the final evaluation of the security and certificate state, but its output is combined with a separate risk assessment framework that considers values of assets and their potential risks. For this reason, the Continuous Certification Evaluation does not deal with the additional risk-driven parameters as proposed in [13], but focuses on effectively aggregating the information received by assessment results.

As mentioned, the problem addressed in [8] and [11] was to rank CSPs according to the CSC's needs. The problem addressed in MEDINA is slightly different, as here the CSP's compliance is determined with a specific standardization (level). In the typical case, all controls and requirements of a standard need to be fulfilled for the cloud service to be (or to remain) certified, although a minor non-conformity occurring for a limited amount of time does not invalidate the certificate. For this reason, it can be useful (for user's review as well as further risk calculation) to observe the level of fulfilment at all layers in the standard's hierarchy, not only the binary information about (non-)conformity.

The methodology used in the Continuous Evaluation component is thus based on building the evaluation tree with assessment results in its leaves, aggregated according to the standard's hierarchy. The aggregation is done with weighted arithmetic means, following the approaches mentioned above. The approach from [11] can be simplified though as assessment results in MEDINA only include binary values (1 meaning conformity and 0 meaning non-conformity), which means that there is no over-provisioning and the masquerading effect does not apply. Additionally, since the goal is to also present intermediate fulfilment values in all levels of the aggregation tree (not only at its root for the entire certification fulfilment), thresholds should be set to determine the fulfilment in individual tree nodes (controls, control groups, etc.). These thresholds and the aggregation weights of the nodes can be set by the user or the auditor (e.g., based on the importance of evaluated resources or controls). The evaluation tree can be easily simplified to an AND tree by setting the thresholds in all nodes to 1, meaning that all the assessment results must indicate fulfilment for the evaluation to be positive, irrespective of the assigned weights (as long as they are positive). The design of the Continuous Evaluation component is further explained in Section 4.

2.2 Digital Audit Trails

MEDINA framework includes digital audit trails as security mechanisms to improve the integrity, traceability and availability of the most relevant information considered in MEDINA (evidence and assessment results). Digital audit trails are detailed and chronological records of important information that are usually used to verify and track all related processes (updates).

Nowadays, audit logs provide a very useful service, allowing auditing processes, secure information storage, tracking of changes made to recorded data (audit trail) and discrepancies, anomalies, and malicious activities detection. However, current audit logs implementations can be vulnerable to a series of attacks, which enable adversaries to tamper data and audit logs, so integrity, which is highly necessary, could be compromised. In addition, audit logs are usually under the control of a central authority which controls and manages information records.

To counter the aforementioned attacks, **Blockchain** technology has started to be considered as a suitable technology for auditing purposes. Although the boom of the Blockchain technology is quite recent, it offers the promise of a **secure, transparent, and affordable solution** to audit trails. On the one hand, Blockchain solves the trust on a central authority problem by maintaining records and transactions of information resources through a distributed network, rather than in a central authority. On the other hand, Blockchain creates an immutable record of transactions; in other words, the need for an immutable audit data storage that is not governed by a central authority can be guaranteed with Blockchain.

In general terms, Blockchain is a Distributed Ledger Technology (DLT) created over a distributed and decentralized network of peer nodes which maintain a copy of the ledger by applying transactions that have been previously validated by a consensus protocol and grouped into blocks with a cryptographic hash that bind each block to the preceding block. This way, given the last block, the previous ones cannot be modified without altering subsequent blocks (i.e., data is practically **resistant to modification**). Another key aspect of these data structures is that transactions are digitally signed by the creator so the **origin** of a piece of data can always be traced back to its creator. Additionally, Blockchain eliminates the need for a central control authority to manage transactions or keep records. The main features for Blockchain technology are:

- **Decentralization:** There is no central authority and no central data storage → no single point of trust, vulnerability or failure.
- **Trustlessness:** Blockchain does not require trust in any authority or participant.
- **Transparency and traceability:** All transactions in Blockchain are visible and verifiable.
- **Immutability:** Transactions and blocks added to the Blockchain are technically impossible to manipulate or modify.
- **Security:** assets in Blockchain are cryptographically secured; due to its decentralization, there is no single point of failure being DoS resistant.

These features are really appreciated by audit trail systems like the one to be included in the MEDINA framework.

2.3 The Cloud Security Certification Life-Cycle

Increasing the degree of automation in the management of certificates requires first to model and then implement the certificate's possible states. The EUCS defines several such states: *Renewed*, *Continued*, *Updated*, *New Certificate*, *Withdrawn*, and *Suspended*. An issuance or state change follows after a review by the CAB. In the literature, different (semi-)automated lifecycle models can be found for cloud security certifications, defining different states and state change procedures.

Kunz and Stephanow [14] define a process model for the continuous certification of cloud services based on two main requirements. First, the target of certification (TOC) may change frequently, so a frequent re-discovery of the TOC needs to be done. This requirement is addressed in MEDINA's WP3. Second, the certificate's state may change any time based on the results of the certification techniques and needs to be reported. They also discuss the implications of automatically reporting certificate updates. Furthermore, they note that several degrees of automation can be targeted in between the traditional, manual process, and the completely automated one. They define three high-level phases for the traditional, manual process, which are derived from several certification standards: *Initialization*, *Audit*, and *Certification*, which are repeated in cycles. Their proposal for a continuous process adds a Scoping phase to define the scope of the service to be audited which includes the discovery of existing cloud resources.

Anisetti et al. [15] propose a semi-automated certification scheme that includes the following phases: *Not Issued*, *Issued*, *Suspended*, *Expired*, and *Revoked*. They furthermore define transition conditions as shown in Figure 1, which presents a finite state automaton. Existing approaches of (semi-)automated certification usually start with an initialisation phase that sets up the necessary tooling, e.g., discovery mechanisms, smart contracts, etc., and aim at verifying the certificate's current state thereafter automatically—or changing it if it doesn't comply with the pre-defined conditions.

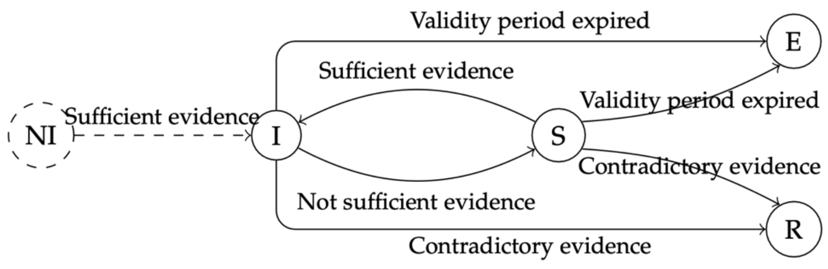

Figure 1. Certificate State-Change Model by Anisetti et al. [15]

In this deliverable, the state model for security certificates is based on the states and change criteria defined in the EUCS [1] and is implemented in a finite state automaton similar to the one described in [15].

3 Architecture

This section describes the overall architecture of the WP4 components. First, the overall goals are explained. Then, the architecture is presented and described. More detailed descriptions of the components and data models are included in the following sections.

3.1 Design Goals

Overall, the goal of Work Package 4 is to process the gathered, and pre-assessed evidence and consequently decide on the certificate state. To that end, several steps are necessary. First, the assessment results need to be aggregated according to their certification requirements. This step needs to be executed continuously since assessment results are generated continuously by the WP3 components as well. Second, the result of this aggregation needs to be enriched using service-specific information. This step is necessary because not all non-compliances are equally severe. Only after this step has been done, an informed decision on the existence of significant deviations can be made, and a translation to a state change can be done.

The components therefore need to process data, like assessment results, continuously. They should also be independently executable to allow for different deployment options. The lifecycle manager, for example, may be deployed by the CSP to manage different certificates; yet also the certification body may use it to manage the state of their customers' certificates. While this work package aims at automating large parts of the certification evaluation process, the developed components should also present a useful means for internal and external auditors, for instance to investigate deviations.

3.2 Architecture Overview and Data Flow Model

Figure 2 shows an overview of the developed architecture which is briefly described in the following. The entry point of the WP4 components is the interface between the Orchestrator (WP3) and the Continuous Certification Evaluation component (CCE). This data flow is designed as a stream of assessment results that are sent to the CCE (Section 4). The CCE in turn aggregates the assessment results to evaluate the overall certificate status on different levels of its hierarchies, e.g., its requirements and controls. The result of this evaluation is an impact-agnostic, tree-based representation of the certificate's compliance state. Only in the next component, the risk assessment, are the results evaluated in more detail considering their individual context, possibly including threat and impact levels. This detailed risk assessment allows then to make an informed decision about the certificate state in the lifecycle manager (Section 6), and to publish the current state.

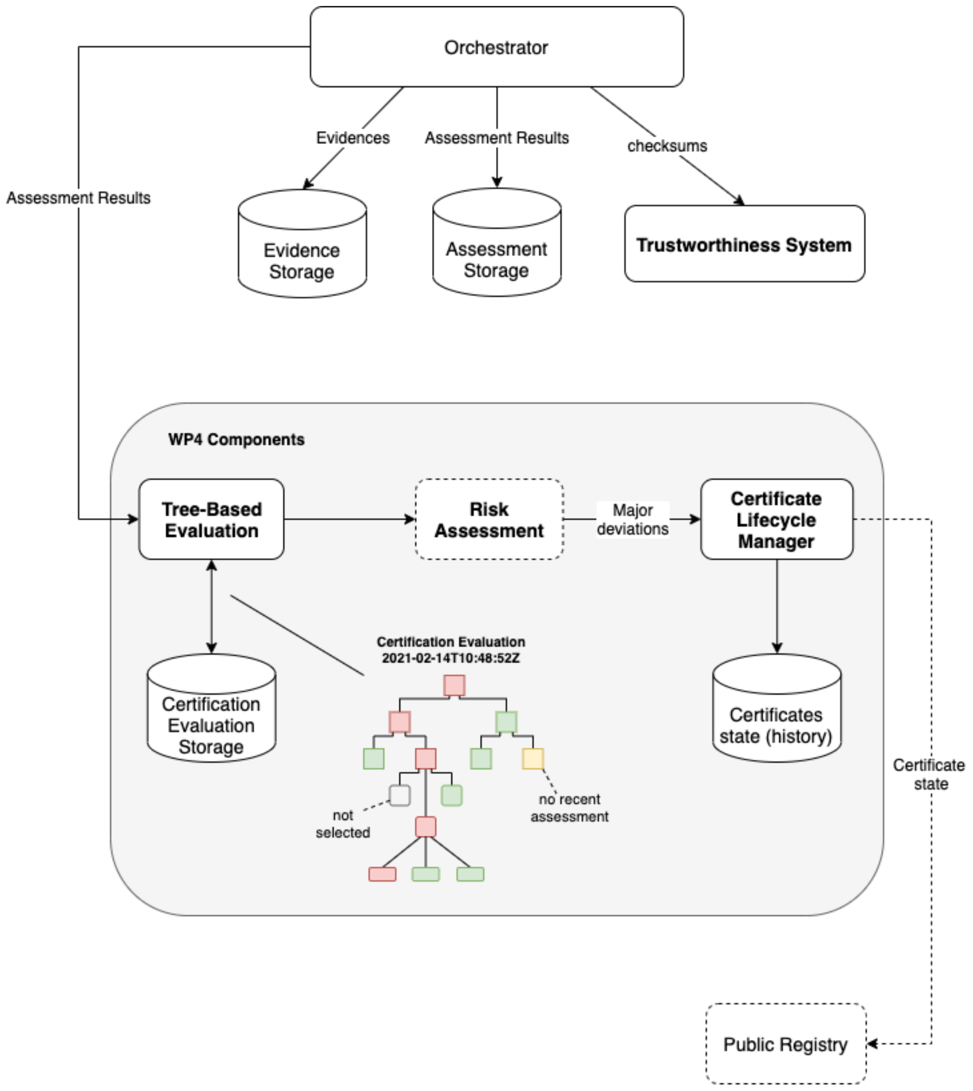

Figure 2. Overall Architecture of WP4 Components, and connection to WP3 components

4 MEDINA Continuous Evaluation of Cloud Security Certification

This section describes the Continuous Certification Evaluation (CCE) component of MEDINA. This component collects assessment results and builds an evaluation tree representing the aggregated assessment results on higher levels of the certification scheme to determine compliance with the different certification elements. The following subsections describe the MEDINA requirements to be fulfilled by this component, its design, and initial information about its implementation.

The first integrated version of the Continuous Evaluation of Cloud Security Certification component has been published as an open-source project[1].

4.1 Approach and Design

4.1.1 Functional Requirements

The following requirements, fully defined in D5.1 [4], are addressed (partly) in the first iteration:

- **CCCE.01:** The evaluation component must be able to evaluate continuously generated evidence and assessment results according to previously defined TOMs to calculate a degree of fulfilment.
- **CCCE.02:** The evaluation component must be able to evaluate continuously generated assessment results according to previously defined TOMs to calculate the degree of fulfilment per individual audited resource and for the TOM in general.
- **CCCE.04:** The evaluation component must provide APIs to the relevant WP3 components to provide measurement results, as well as to the digital audit trail and the certificate lifecycle management component to exchange relevant data.

4.1.2 Interaction with Other Components

The data flow from gathering of technical and organizational evidence to the certificate lifecycle management is represented in Figure 3 below, showing the Continuous Certification Evaluation in relation to the other components. Assessment results originating in the Security assessment component(s) are being forwarded to the Certification evaluation component through the Orchestrator. A single assessment result object contains an assessed value of a specific metric (whether it is fulfilled or not) for a specific resource of the CSP's infrastructure.

As described in Section 4.1.3, the Continuous Certification Evaluation aggregates this information into an evaluation tree, which is stored in the Certification evaluation storage database and forwarded to the Risk Assessment component to further evaluate the non-conformities detected and drive the Automated certificate management. At this point in the project, it is not yet clear whether the Risk Assessment component will consume the entire evaluation tree from the Continuous Evaluation component or only a part of the tree or individual non-conforming nodes. The risk assessment details are to be defined in Task 4.4, starting in month 12 of the project.

The Continuous Certification Evaluation component is also linked with the Catalogue of controls and security schemes (developed in WP2), from which it gathers the structure of the used certification scheme (lists and mappings of metrics, requirements, controls, control groups…), needed to construct the evaluation tree. The catalogue may also provide a list of monitored resources.

[1] https://git.code.tecnalia.com/medina/public/continuous-certification-evaluation

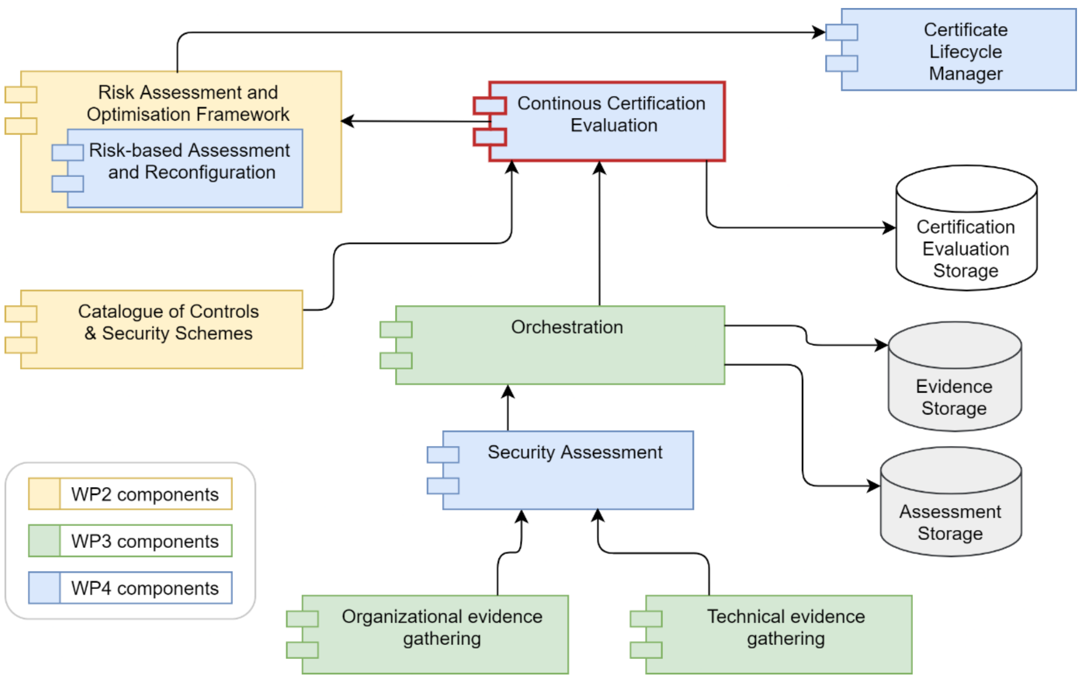
Certificate Lifecycle Manager
Risk Assessment and Optimisation Framework
Risk-based Assessment and Reconfiguration
Continous Certification Evaluation
Certification Evaluation Storage
Catalogue of Controls & Security Schemes
Orchestration
Evidence Storage
Security Assessment
Assessment Storage
WP2 components
WP3 components
WP4 components
Organizational evidence gathering
Technical evidence gathering

Figure 3. Continuous Certification Evaluation: diagram of interaction with related components (adopted from D5.1 [4])

4.1.3 Certification Evaluation Methodology

As explained in Section 2.1, the method for aggregation of assessment results in the Continuous Certification Evaluation component follows the tree-like hierarchy of the various standardisation schemes, like shown on Figure 5. Values in the tree are evaluated bottom-up: from the leaves that represent assessment results to the root representing the complete certification scheme and thus indicating the fulfilment of the certificate.

The general design of the Evaluation component is made to be modular and adaptable in terms of aggregation and tree building. The aggregation can be made with various methods, also by combining different methods at different levels of the tree. Building of the tree skeleton can be made in advance if all the relevant resources and their mappings to requirements and metrics are known before gathering the evidence. Alternatively, the tree structure can be built while receiving assessment results and discovering the resources and the requirements that they must fulfil.

Described below is the current proposal of the methodology, which will be refined in the coming months in collaboration with and according to the needs of risk assessment activities (Task 4.4).

4.1.3.1 Building the tree structure

The evaluation tree (see Figure 5) is logically composed of two parts: in the upper part, the structure is defined directly by the standardisation scheme being used: control groups, controls, and requirements. There is a possibility that controls can be selected or unselected by the user if allowed by the standardisation scheme in use. The levels below the level of requirements are not directly defined by the standard but are important to determine the compliance values of elements higher in the hierarchy. The conformity to a requirement is determined by measuring one or more metrics related to this requirement, and there can be multiple resources on which the measurements are made. A single assessment result contains the information about whether a particular monitored resource conforms to the target value for a specific metric. To use the assessment results for computing the conformity values of requirements, three aggregation techniques are described below:

a) directly aggregating assessment results into compliance values of requirements,
b) combining assessment results of different resources into compliance values of metrics, and combining metrics into compliance values of requirements,
c) combining assessment results of different metrics into compliance values of resources and combining resources into compliance values of requirements.

The above-mentioned techniques are represented graphically in Figure 4. Technique a) is the simplest, avoiding the additional aggregation layer. The downside of this approach is the lack of visibility of metrics' or resources' compliance levels – the compliance levels of resources or metrics are not computed and cannot be examined by users. Also, aggregation weights of metrics and resources cannot be assigned individually but have to be combined into a single value.

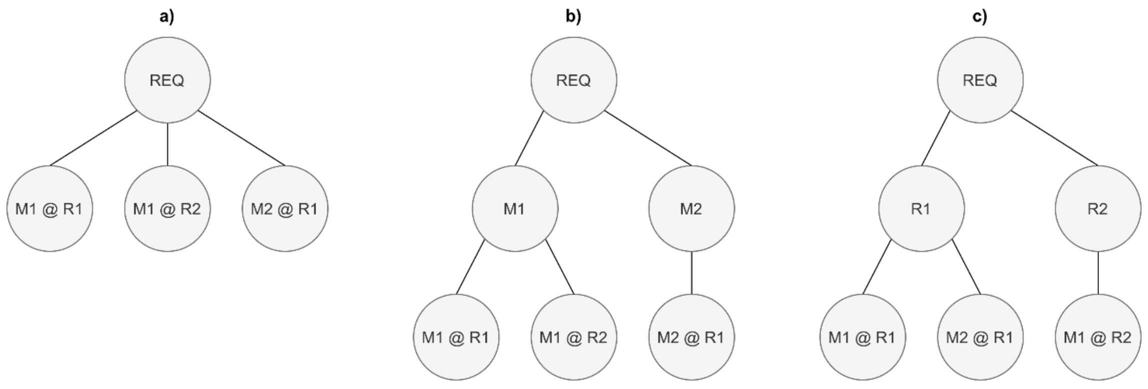

Figure 4. Possible options for aggregation of assessment results into compliance levels of requirements

The difference between options b) and c) is whether assessment results are aggregated into compliance values of metrics or into compliance values of resources, respectively. Technique b) calculates whether some metric is satisfied across all relevant resources, whereas option c) evaluates whether some resource is satisfying the related requirement considering all relevant metrics. The aggregated value at the requirement level will be the same in both options b) or c) in case when values of all metrics under the requirement in question have been measured for all relevant resources. When this is not the case (example shown in Figure 4 and described below), the requirement value computed using technique b) is affected greater by non-conformities in assessment results of metrics, measured on fewer resources, while with technique c) assessment results of resources where fewer metrics are measured are regarded as more important (considering similar aggregation weights).

Let us consider the example shown in Figure 4, where both metrics 1 and 2 are evaluated for resource 1, but only metric 1 is evaluated for resource 2 (no assessment result was obtained for metric 2 on resource 2). For this example, we assume that weights for all resources and all metrics are the same. In case all assessment results are positive, the requirement value is 1 for all methods. Table 1 shows a comparison of the calculated fulfilment value for a requirement when one of the assessment results is negative (the other two are assessed as positive) with different methods of aggregation. Method a) considers all three assessment results equally, thus the requirement value is $^2/_3$, regardless of which single assessment result is negative.

When metric 1 at resource 1 is evaluated negatively, both b) and c) methods return the same requirement value since (as evident from Figure 4) this assessment result is represented as half of metric 1 (b) and also as half of resource 1 (c). In case when metric 2 at resource 1 is evaluated negatively, method b) returns a requirement value of 0.5, while it is 0.75 with method c). Method b) penalizes this case harder because this is the single assessment result for metric 2 – one of the two metrics in the second tree level is evaluated with 0. Analogously, when metric 1 for resource 2 is non-conformant, it is penalized harder with method c) since this is the only metric evaluated on resource 2.

The last column in the table shows the c) method of aggregation where the aggregation results for different metrics are aggregated with the AND approach – each resource is assigned a Boolean fulfilment value: 1 if and only if all metrics for this resource are evaluated positively; 0 otherwise. With this method, the requirement values for all cases in our example are 0.5 because one of the two resources on the second tree level is evaluated negatively in each case. Metric aggregation using AND rule is further discussed below.

Table 1. Comparison of requirement fulfilment value depending on non-conformity of individual assessment results calculated with different aggregation methods

	a)	b)	c)	c), AND metric aggregation
M1 @ R1 negative	0.67	0.75	0.75	0.5
M2 @ R1 negative	0.67	0.5	0.75	0.5
M1 @ R2 negative	0.67	0.75	0.5	0.5

If the evaluation tree is built using technique b), the users are able to see the conformity level by metrics and, if needed, they can examine the individual metrics to discover which resources under this metric are contributing to some non-conformity. Alternatively, with option c), users can see the conformity levels of their resources with all metrics linked to a requirement aggregated. They can examine the lowest tree level (metrics) to determine which metrics in that resource are problematic. Due to this, we believe that most users would find option c) more informative.

As explained below in Section 4.1.3.2 below, the approach for the initial MEDINA proof-of-concept considers that all metrics for a particular resource need to be evaluated positively to regard the requirement fulfilled. Aggregation of the metrics level is thus made with simple AND rules and weighted aggregation does not apply at this level. On the other hand, configuration of different weights for resources can be desirable from the risk assessment perspective. With aggregation technique b), fulfilment values of metrics are calculated from multiple resources and are thus not Boolean values. If we are to apply AND aggregation on metrics, we could consider the metrics values positive or negative depending on thresholds. Regardless of thresholds though, the weights of resources used on the leaf-level would become irrelevant at the requirement and higher levels of the evaluation tree (Boolean fulfilment values are applied to requirements). With technique c), the assessment results are aggregated into resources' compliance levels using AND, applying Boolean values to resources. The compliance values of resources can therefore be aggregated into requirements' fulfilment levels using their respective weights.

Following the considerations described above, technique c) was chosen for the initial implementation of the Continuous Certificate Evaluation component and is therefore considered in the following description of the tree-building process. As an example, a part of such evaluation tree is also shown in Figure 5. As mentioned, the component is implemented in an adaptable way, meaning that if additional requirements are found, refinements of the approach are possible and would not require significant effort.

The first proof-of-concept implementation of the MEDINA framework does not plan to include a repository of all monitored resources related to the evaluated cloud service. For this reason, the entire evaluation tree structure cannot be built in advance (before receiving the assessment results for individual resources). In the start-up phase of the component, the tree structure is built down to the level of requirements by obtaining the elements of the certification scheme and the mappings between the hierarchy levels from the Catalogue of controls and security schemes. The lower part of the tree is built part by part during the component's operation.

When receiving an assessment result for metric *M* and resource *R*, the component first checks whether such an assessment result is already present in the evaluation tree. In this case, its values (there can be multiple tree nodes corresponding to an assessment result when a single metric maps to several requirements) can simply be updated and propagated to the higher hierarchy levels through aggregation. If no nodes with metric *M* and resource *R* exist, they need to be added to the tree. Resource *R* is added as a child node to all requirements that metric *M* is associated with. For all such added nodes of resource *R*, metrics that are required for fulfilment of particular requirements are added as child nodes representing assessment results. The values of these assessment result nodes remain undefined (except the assessment result received for metric *M*) until a matching assessment result is received.

4.1.3.2 Aggregating the evaluation values

While different aggregation methods can be used for calculating the compliance values in the evaluation tree, the main method proposed in the initial proof of concept is setting the value of a node with a weighted arithmetic sum of the child nodes' values. The reason for choosing this approach is explained in Section 2.1. As shown in Figure 5, each tree node (representing an element in the standardisation hierarchy) has two configurable parameters: weight w and threshold T, and its value V is calculated using the weighted average of its child nodes:

$$V = \frac{\sum V_i w_i}{\sum w_i}$$

where i runs across the child nodes. Since the weighted sum is divided by the sum of weights, node values (and, consequently, thresholds) always fall in the interval $[0,1]$.

Thresholds simply mark the (un)conformity of a node by regarding nodes with $V \geq T$ as compliant. In the current proof-of-concept, thresholds are used mostly for visibility, to clearly display the nodes' (un)conformities to the user and to trigger the additional risk assessment evaluation of non-conformities. Another option would be to regard the nodes' values in their aggregation on the parent level as totally (un)compliant (0 or 1) depending on their compliancy with respect to the threshold. This way, the weighted aggregations would not propagate further than one level in the tree.

The evaluation tree can be easily simplified to an AND tree by setting all threshold values to 1.

The leaf nodes (representing assessment results) are expected to have logical Boolean values (evaluated by the Security Assessment components with respect to the evidence's compliance with the metric's target value), meaning that their values can only be 0, 1, or undefined (in case where no assessment results have been obtained for a specific metric-resource pair). Undefined values are regarded as uncompliant (0). As already mentioned, MEDINA defines metrics related to a particular requirement of the standard as a set of constraints which all need to be fulfilled to regard the requirement as compliant. For this reason, aggregation on the first level of the evaluation tree (from assessment results to compliance values of resources for a specific requirement) is done using the AND approach – resource nodes are assigned a value of 1 only if all metrics for a requirement are satisfied (or 0 otherwise).

Weights of individual elements can be assigned by the CSP (in collaboration with the auditor) and possibly with inputs from the risk management framework according to the CSP's risk appetite.

If allowed by the specific standardisation scheme and chosen by the CSP (as well with inputs from the risk management), some elements of the scheme (nodes of evaluation tree) can be disregarded in the evaluation. In the example shown below (Figure 5), one control of the standard is not selected and thus ignored in the aggregation to its parent node (control group).

Above we presented different methods that can be used in the Certification Evaluation Component in order to support various standards and certification schemes. Details of the actual implementation of the component in MEDINA are to be refined and agreed in collaboration with other tasks, especially according to the risk assessment methodology used in Task 4.4.

4.2 Implementation

At the current time, an initial prototype is implemented with basic tree building and aggregation functionalities and no user interfaces or APIs added. The communication format with the Orchestrator component (WP3) is defined according to the general MEDINA data model and gRPC as communication protocol. Details about the communication with other required components (WP2 Catalogue of Controls and Security Schemes and WP4 Risk Assessment component) are currently being defined. As explained above, some details about the aggregation and evaluation methodology are dependent on the risk assessment methodology being currently developed.

The programming language used for the Continuous Certification Component is Java.

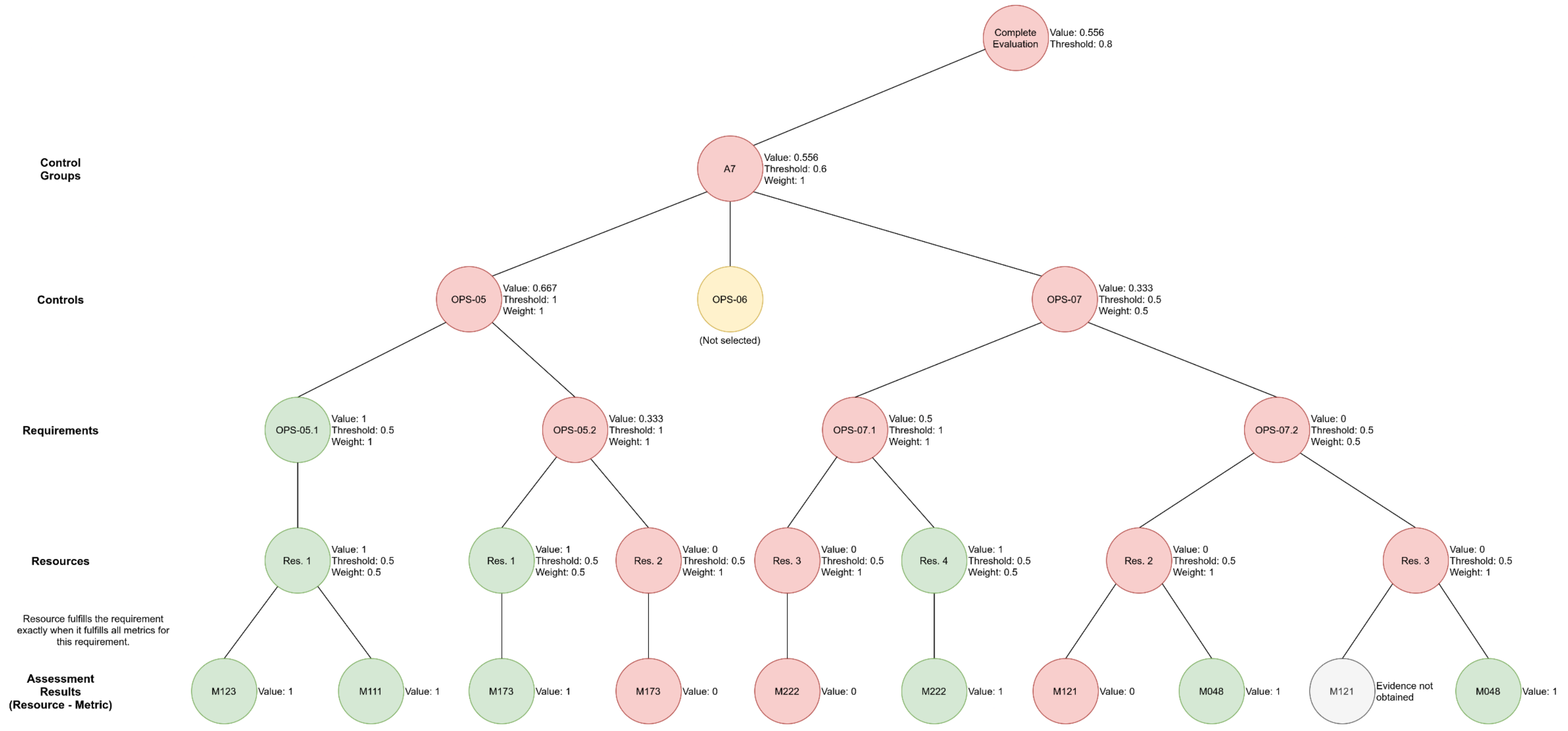

Figure 5. Example of (a part of) an evaluation tree representing (non-)conformities of standardisation hierarchy elements

4.3 Limitations and Future Work

The evaluation tree built by the CCE component is an enhanced representation of data coming from the evidence gathering and security assessment tools. The confidence of the CCE's outputs thus largely depends on the data provided by those components.

The CCE can be efficiently used to review the state of gathered evidence at some point in time, but a limitation is that no conclusions about the actual risk state or the certification status can be made solely based on the CCE outputs. Other components of the MEDINA solution (Risk Assessment and Optimisation Framework and Certificate Lifecycle Manager) help users understand the broader view of their certification state.

Due to the missing functionalities in the current version of the CCE (at month 12 of the project), some temporary limitations exist. History and statistics of evaluation states cannot be displayed currently, the integration with other components is not fully functional, the authorization and authentication systems are missing, and the user interface only supports the basic view of the current evaluation tree status.

Consequently, future work for this component will address improvements in the integration with other components, UI enhancements, and will explore possibilities to leverage the tree structure that the CCE builds. For example, interesting metrics can be applied regarding the fulfilment of certain requirements over time.

5 MEDINA Establishment of a Digital Audit Trail

This section will theoretically identify the need of a digital audit trail in MEDINA for increasing the trustworthiness of the overall framework by means of a risk assessment analysis. It will also analyse different existing technologies that could be considered for a digital audit trail with a clear focus on the Blockchain technology, which greatly improves current digital audit trail implementations.

5.1 Risk Assessment

MEDINA framework has been identified as a potential high-risk system as it operates over sensitive information, such as evidence and assessment results of critical resources.

The purpose of this risk assessment is to identify the threats and vulnerabilities, and to identify different ways to mitigate those risks. The risks are assessed following a standard methodology (see e.g., Torr [16], or the NIST guidelines [17]). First, we state assumptions regarding the MEDINA framework. Then we identify the assets to be protected, their protection goals, as well as the users of the system. We continue by defining the attacker model and then model possible attack vectors. Risks are then assessed based on the likelihood of an attack and their potential impact. Finally, we discuss the possible mitigations.

This risk analysis is central to the effective implementation of a trustworthiness system inside the MEDINA framework.

5.1.1 Assumptions

It is important to highlight some assumptions to be considered for this risk analysis. In this case, all MEDINA tools are considered reliable**; we trust all MEDINA tools**. As a result, MEDINA tools cannot be considered as threats nor any vulnerability could be detected in their implementation and/or operation.

Furthermore, human factor vulnerabilities are not considered in this analysis as they are complex, undefined, non-linear, and often not repeatable in a predictable way. This is the case of social engineering-based vulnerabilities.

5.1.2 Asset classification scheme

The first step in a risk assessment process is to identify and define all valuable assets in scope. This risk analysis is focused on **critical data**, or other data whose exposure would have a major impact on the MEDINA framework operation.

Table 2. Overview of types of data and their sensitivity levels

Type of data	Description	Level of sensitivity
Evidence (for more details, see trustworthy evidence data model)	• id • toolId • resourceId • cspid • measurementResult • timestamp	• The field *measurementResult* has a high level of sensitivity. • The rest of the fields has a low level of sensitivity.
Assessment Results	• Id • metricId • assessmentResult • complianceResult	• The fields *assessmentResult* and *complianceResult* have a high level of sensitivity.

Type of data	Description	Level of sensitivity
	• associatedEvidencesId • timestamp	• The rest of the fields have a low level of sensitivity.

5.1.3 Potential users

Next, it is recommended to identify and describe who is going to operate or access the assets identified in section 5.1.2 as critical data.

Table 3. Overview of the different users

User	Data	Access Level	Number of users	Organization
Orchestrator	• Evidence • Assessment Results	Full (Read and Write)	One instance per organization	Internal organization
Organization employees	• Evidence • Assessment Results	Read only	Undetermined	Internal Organization
Auditors	• Evidence • Assessment Results	Read only	Undetermined	CAB

5.1.4 Protection goals

Information assurance is an approach of managing risks related to the use, processing, storage, and transmission of information or data. The three main components and goals are to protect and ensure the confidentiality, integrity, and availability (CIA) of information. However, some additions are also relevant for guaranteeing information security: authenticity, authorization and non-repudiation.

Confidentiality

Confidentiality is the property that guarantees information is not made available or disclosed to unauthorized individuals. Assets gathered in section 5.1.2 are sensitive information about specific organizations that should be kept private from all unauthorised users; confidential information must only be accessed by authorized users; in MEDINA, authorized users are those gathered in section 5.1.3. **In MEDINA, confidentiality is a must**, since evidence and assessment results contain sensitive information about the security posture of the audited service provider.

A confidentiality breach occurs if unauthorized people or systems access information they are not allowed to, e.g., when an attacker eavesdrops on unencrypted communication channels.

Integrity

Integrity is the property of safeguarding the accuracy and consistency of assets; it means that information cannot be altered or tampered with, ensuring the data correctness and protecting against unauthorized modification. In MEDINA, auditors need to trust the stored data regarding its integrity to provide corresponding certificates. For that reason, **in MEDINA, integrity is a must**.

An integrity loss occurs if shared information is somehow modified, causing the information to become unreliable and, consequently, audits using MEDINA framework could not be trusted.

Availability

Availability is the property of being accessible and usable upon demand. Availability assumes that information systems, as well as the information itself, is available and operating as expected when needed or requested. In MEDINA, evidence and assessment results should be available for the proper orchestrator operation as well as for auditors to verify them when needed. Consequently, **although it is not a must in MEDINA, it is highly recommended to guarantee evidence and assessment results availability**.

An availability loss means the stop of the proper system operation or data access for a significant length of time (usually several minutes), limiting the MEDINA framework operation or delaying the access to the required information by auditors.

Authenticity

Authenticity is the property that guarantees an entity is what it claims to be, proving that all parties involved in an action are who they claim to be. It is of great importance to ensure the genuineness of every asset, reducing instances of fraud by way of misrepresentation.

MEDINA needs to authenticate all the information sources in order to certify who provided, modified or even deleted certain data related to evidence and/or assessment results. By this way, auditors will be sure that trusted sources have operated the MEDINA framework and no impostor source has ever replaced legitimate sources. **In MEDINA, authenticity is a must.** However, due to the assumption based on trusting all the MEDINA tools, authenticity of evidence and assessment results is given. Anyway, some additional secure authentication mechanisms, such as mutual authentication between different MEDINA components, could be added so that anyone outside the MEDINA system could provide information.

Authorization

Authorization is the property that determines access levels or user privileges related to system resources including information. It is related to the access control techniques, granting or denying access to a specific resource depending on the user identity. **This is not the role of MEDINA** as MEDINA is a tool developed and used by auditors, with the same access levels or user privileges. All potential MEDINA users will have the same role: auditors.

Non-repudiation

Non-repudiation is the ability to prove an event or action has occurred as well as to identify its originating entities in order to resolve disputes about the occurrence or non-occurrence of the event and who were the involved entities.

In MEDINA, non-repudiation is very relevant in two senses. On the one hand, sources providing data (evidence and/or assessment results) should not be able to deny their involvement in MEDINA; once they provide data, they cannot deny their data provision. However, due to the assumption based on trusting all the MEDINA tools, non-repudiation is already guaranteed. On the other hand, sources accessing data (evidence and/or assessment results) should neither be able to deny their involvement in MEDINA; once they access data, they cannot deny the have read the data.

5.1.5 Potential attackers

It is important to develop a catalogue of potential attackers, in other words, threat sources. There are two main types of attackers: outsiders and insiders.

In general, **outsiders** can be classified based on their professional level: organized attackers, hackers and amateurs.

- Organized attackers (terrorists, nation states, and criminals). They are generally highly trained, highly funded, tightly organized, and are often backed by substantial scientific capabilities. In many cases, their highly sophisticated attacks are directed toward specific goals.
- Hackers: they may be perceived as benign explorers, malicious intruders, or computer trespassers. In most cases, they are highly trained and could be sponsored by criminal organization or governments for financial gain or political purpose, increasing their financial capability.
- Amateurs: these are less-skilled hackers, also known as "script kiddies" who often use existing tools and instructions that can be found on the Internet. They are not as dangerous as the previous ones since they do not have the ability to create their own, adapted tools.

In general, **insiders** are people from the own organization (or with a strong relation with the organization) who have skills, knowledge, resources, and access to the organization systems. Consequently, malicious insiders will have a deep knowledge of the MEDINA framework.

It is recommended to identify threat actions that could negatively affect the MEDINA framework operation regarding the identified protection goals and attacker types, from security breaches to human errors.

Table 4. Overview of main potential threats from different attackers

Attacker	Threat Action
Outsiders	• System intrusions • Identity theft
Malicious insider	• Browsing of personally identifiable information. • Unauthorized system access through escalation of privilege. • Accidental or ill-advised data modification/deletion • Accidental or ill-advised actions taken by employees that result in unintended physical damage, system disruption, etc.
Environmental	• Natural or man-made disasters; HW failure, etc.

In addition, it is recommended to identify potential attackers' motivations in order to determine the real risk, as it is not the same a threat for political reasons than a threat from an accident of a trusted employee.

Table 5. Overview of main motivations for different attackers

Attacker	Motivation
Outsiders	• Someone who wants to change data to ensure the certificate is not obtained by the organization.

Attacker	Motivation
	• Someone who wants to obtain sensitive information (e.g., for espionage). • Unhappy customers who want to damage the organization (discredit, loss of customers, etc.). • Intellectual challenge. • Social/political/economic incentive.
Malicious insider	• Someone who wants to change data to successfully obtain a certificate. • Unhappy workers who want to damage the organization (discredit, loss of customers, etc.). • Someone who makes a mistake modifying or deleting information (trusted employees accidentally misplacing information).
Environmental	• N/A

5.1.6 Potential attacks

It is essential to assess which vulnerabilities and weaknesses could allow potential attacks breaching the MEDINA framework security.

Table 6. Description of the main potential attacks in MEDINA

Protection goal	Potential attack	Description
Confidentiality	• Eavesdrop on database connection • Eavesdrop on tool connection	Secretly listen to the private communication between the gathering/assessment tools and the orchestrator and between the orchestrator and the database without consent to gather data (or metadata) information. It is usually related to a lack of encryption services.
	Gain read access to database	Broken access control vulnerabilities exist when a user can access specific data that they are not supposed to be able to access. It is related to not enforcing any protection over sensitive data or by means of privilege scalation.
	Phishing	Obtain authentication data by impersonating oneself as a trustworthy entity in order to gain access to private data.
Integrity	• MitM attack on database connection • MitM attack on tool connection	The attacker secretly relays and alters the information in the communication between the gathering/assessment tools and the orchestrator and between the orchestrator and the database who believe that they are directly communicating with each other.

Protection goal	Potential attack	Description
	Gain write access to database	Broken access control vulnerabilities exist when a user can access specific data that they are not supposed to be able to access. It is related to not enforcing any protection over sensitive data or by means of privilege escalation.
Availability	DoS attack to the database	Flooding the database with traffic or sending it information that triggers a crash in order to shut down the system, making it inaccessible to its users. There is a special risk with centralized systems (Single point of failure).
	Internet access down	Internet outage due to an external problem (natural disaster, etc.)
	Gain write access to database	With write access to the database, an attacker can simply delete evidence and assessment results (see the integrity threat).
Authenticity	Phishing for private key (credentials) for database access theft	Obtain authentication data by impersonating oneself as a trustworthy entity in order to gain access to private data.
	Poor private key (credentials) strength for database access	Passwords used are weak. Attackers could guess the password of a user to gain access to the database.
	• MitM attack on database connection • MitM attack on tool connection	The attacker secretly relays and alters the information in the communication between the gathering/assessment tools and the orchestrator and between the orchestrator and the database who believe that they are directly communicating with each other.
Non-repudiation	Phishing for private key (credentials) for database access theft	Obtain authentication data by impersonating oneself as a trustworthy entity in order to gain access to private data.
	Poor private key (credentials) strength for database access	Passwords used are weak. Attackers could guess the password of a user to gain access to the database.
	• MitM attack on database connection • MitM attack on tool connection	The attacker secretly relays and alters the information in the communication between the gathering/assessment tools and the orchestrator and between the orchestrator and the database who believe that they are directly communicating with each other.

5.1.7 Likelihood of Exploitation

The next step involves determining the likelihood of the potential attacks identified in section 5.1.6 resulting in succeeding against our system. Likelihood is the probability that a vulnerability is exercised in an attack. It mainly depends on:

- Attackers' motivation and capacity
- Nature of the vulnerability
- Existence of countermeasures
- History

Probability can be ranked as:

- **High**: the attacker is highly motivated and sufficiently capable; controls to prevent the vulnerability to being exercised are inefficient.
- **Medium**: the attacker is motivated and capable, but controls are in place that may impede successful exercise of the vulnerability.
- **Low**: the attacker lacks motivation and/or capability, or controls are in place to prevent or, at least, significantly impede the vulnerability for being exercised.

Table 7. Likelihood of different attacks to happen

Potential attack	Likelihood
Eavesdrop on database connection	Medium
Eavesdrop on tool connection	Medium
Gain read access to database	High
Phishing	Medium
MitM attack on database connection	Medium
MitM attack on tool connection	Medium
Gain write access to database	High
DoS attack to the database	High
Internet access down	Low
Phishing for private key (credentials) for database access theft	Medium
Poor private key (credentials) strength for database access	High

5.1.8 Impact

The next step in a risk analysis is to perform a risk impact analysis to understand the consequences of an incident. The impact will be used to calculate and prioritize risks in the final step.

- **High impact**: There is a strong need for corrective measures.
- **Moderate impact**: Corrective actions are needed, and a plan must be developed to incorporate these actions within a reasonable period of time.

- **Low impact**: It must be determined whether corrective actions are still required or decide to accept the risk.

Table 8. Overview of effect and impact of the potential attacks

Effect	Impact
Evidence/Assessment results will not be trustworthy	High
Evidence/Assessment results will not be available	Moderate
Audit will not be trustworthy	High
Organization discredit	High
Unfair competence	High

5.1.9 Risk Calculation

The last step in a risk assessment is to combine the likelihood from section 5.1.7 and the impact values calculated in section 5.1.8 to arrive at a risk value. The risk value for the potential attacks is:

Table 9. Overview of the risk of the potential attacks

Potential attack	Likelihood	Impact	Risk
Eavesdrop on database connection	Medium	Moderate	Moderate
Eavesdrop on tool connection	Medium	Moderate	Moderate
Gain read access to database	High	Moderate	Moderate
Phishing	Medium	Moderate	Moderate
MitM attack on database connection	Medium	High	High
MitM attack on tool connection	Medium	High	High
Gain write access to database	High	High	Very High
DoS attack to the database	High	High	Very High
Internet access down	Low	High	Moderate
Phishing for private key (credentials) for database access theft	Medium	High	High
Poor private key (credentials) strength for database access	High	High	Very High

5.1.10 Security Requirements

Based on the risk analysis, mitigation techniques are needed to reduce or even mitigate the identified risks.

- A set of rules are required to be applied to limit access to personal data only to authorized people → User access control: identification & authorization.
- Therefore, data should be kept secure applying **Privacy Enhancing Technologies** (e.g., encryption, pseudonymization, anonymization, identity and access management).
- A set of rules are required to ensure the data is trustworthy and accurate.
- A set of rules are required to prevent accidental disclosure of sensitive data.

Taking these security requirements into consideration, **a secure MEDINA trustworthiness system for audit trail will be included in the MEDINA framework**.

5.2 Audit trail

As it has been presented in Section 2.2, Blockchain technology has started to be considered as a suitable technology for auditing purposes due to some of its main features: decentralization, trustlessness, transparency, traceability, immutability and security. However, other options, such as traditional databases or replicated databases could be also considered.

This section will theoretically compare the three mentioned alternatives, identifying the main advantages and disadvantages in each case. It will also analyse different Blockchain technologies in order to identify the one that better fits MEDINA.

5.2.1 Blockchain vs Traditional databases

At first glance, Blockchain and traditional databases can be considered similar, as both are used, broadly speaking, to store information in a distributed or centralized way. However, **Blockchain is more than just a database**. There are several differences between both technologies:

- **AUTHORITY:**
  - Blockchain: It is decentralized; no central control
  - Database: It is centralized; It is controlled by an administrator

In Blockchain, each node takes part in a consensus mechanism to check all transactions, with the same level of access and capability, democratizing the whole system. In a traditional database, a central authority (administrator) controls the whole system. In this context, Blockchain takes advantage over traditional databases since trust is not required, it is given by design.

- **ARCHITECTURE:**
  - Blockchain: Distributed
  - Database: Client-server architecture

In Blockchain, data is distributed among all nodes; each node stores a copy of the complete Blockchain so although some node is compromised, the rest can continue working. Therefore, single point of failure attacks are infeasible in Blockchain, gaining in robustness and fault tolerance over traditional databases where data is centrally stored in a server.

- **DATA HANDLING:**
  - Blockchain: Read and Write
  - Database: CRUD (Create, Read, Update and Delete)

Traditional databases provide additional functionalities over Blockchain (update and delete). One of the most important features is the ability to delete information. In Blockchain nothing can be deleted; any data included in the Blockchain will be recorded forever.

In the MEDINA context, it is not really needed to update evidence and assessment results in the audit trail, as an update in any of them is considered new evidence or a new assessment result (with a different identifier). In addition, being able to delete existing information is not a requirement for MEDINA.

- **INTEGRITY:**
  - Blockchain: Supported
  - Database: Malicious actors can modify database data

One of the most important features of Blockchain is integrity, which is achieved by the consensus mechanism in which all the participating nodes check and validate all the transactions, and the data distribution among all nodes with every node storing a copy of the complete Blockchain, so it seems impossible for an individual malicious actor to modify the stored information or to include incorrect information in the Blockchain. In traditional databases, on the contrary, the system is vulnerable if the administrator is compromised; as it is a centralized party, it is more likely to happen.

- **TRANSPARENCY:**
  - Blockchain: Supported
  - Database: The administrator is who decides which data can be accessed

In Blockchain, every participating node has the same level of access and capability, creating a totally democratized system in which transparency is guaranteed by design. On the contrary, in traditional databases, the administrator decides who can access the database and what actions can execute.

- **IMPLEMENTATION AND MAINTAINANCE COSTS:**
  - Blockchain: High
  - Database: Low

Blockchain is still a new technology so, generally speaking, its implementation and maintenance is still more costly than for traditional databases based on "old" technologies. However, some existing Blockchain technologies are already widespread, and their costs are beginning to be reduced.

- **PERFORMANCE:**
  - Blockchain: Low (due to the verification and consensus methods)
  - Database: Fast and with high scalability

Traditional databases are known for faster execution time and can handle millions of data at any given time. However, Blockchain is considerably slower because of carrying more operations. including signature verifications and consensus mechanisms. However, in the MEDINA context, a high performance in the audit trail is not a requirement.

5.2.2 Blockchain vs Replicated databases

Some of the traditional databases main disadvantages can be improved by means of replication techniques, copying data from one database to another resulting in a distributed database system with all databases with the same level of information. However, Blockchain still differs from replicated database in the following aspects:

- **REPLICATION:**
  - Blockchain: Transaction replication
  - Replicated Database: State replication

Blockchain replicates entire transactions so that its execution can be replayed by each participant node. Distributed databases, on the contrary, replicate the resulting log of read and write operations. For this purpose, a distributed database management system is needed to guarantee that updates, additions and deletions performed on the data at any given database are automatically reflected in the data stored at all the other databases. So, as the administrator of a traditional database, it is a centralized party that must be trusted. In contrast, Blockchain does not need any trusted entity, therefore it replicates the entire transactions so that their execution is replayed by each participating node.

- **CONCURRENCY:**
    - Blockchain: No (serial execution)
    - Replicated Database: Yes

Most Blockchains support only serial execution as the transaction execution is not the real bottleneck in the Blockchain performance (the consensus mechanism usually is) and, by this way, the behaviour of smart contracts is deterministic when the transaction execution is replicated over many nodes, being easier to identify the ledger states. Distributed databases, on the contrary, employ sophisticated concurrency control mechanisms to extract as much concurrency as possible and improve performance.

Although concurrency is not a real concern in MEDINA, some recent Blockchains have started to adopt some simple concurrency techniques, such as, for example, in *Hyperledger Fabric* where transactions are executed in parallel against the ledger states before being sent for ordering.

5.2.3 Blockchains technologies

5.2.3.1 About consensus algorithms

First of all, it is not possible to compare different Blockchain technologies without introducing some of the most famous consensus algorithms. All technologies use their own consensus algorithm or a combination of some of them. Some of the most used consensus algorithms are described below.

- **Proof-of-Work (PoW)**

Proof-of-work based consensus mechanisms require the resolution of a computationally expensive calculation to validate a block. Mining nodes (the nodes in the network responsible for validating or "mining" blocks) compete to solve the computation and mine the block and are rewarded with a fee. It was the first consensus mechanism used in Blockchain (it is the one used by *Bitcoin*), although alternatives have been emerging with the purpose of improving some of its aspects (energy consumption, risk of network centralization, etc.).

- **Proof-of-Stake (PoS)**

In this case, the node creating the block is selected deterministically. The richness (number of tokens accumulated by a node) of each node is positively involved in this selection. The main problem of this "nothing-at-stake" consensus algorithm is that when a chain is split, since it costs nothing, the two forks are bet on. This makes it so that consensus on a single Blockchain is not guaranteed. It is used by *Nxtcoin*, *Peercoin* or *Bitshares*, for example. Ethereum has decided to incorporate Proof of Stake by means of the Casper Protocol.

- **Proof of Authority (PoA)**

PoA is a modified form of PoS where instead of stake with the monetary value, a validator’s identity performs the role of stake. In this context, identity means the correspondence between a validator's personal identification on the platform with officially issued documentation for the same person, i.e., certainty that a validator is exactly who that person represents to be. Just like in PoS, in PoA consensus, identity as a form of stake is also scarce. But unlike PoS, there's only one identity per person. *Kovan* and *Rinkeby*, the two Ethereum test nets, use PoA.

- **Casper protocol**

Casper emerges as a hybrid between PoW and PoS and it is currently the algorithm that Ethereum is trying to implement. That is why it is actually considered by some authors as a PoS type algorithm.

Casper works as a kind of wager in which different nodes propose blocks that should be added to the chain. The validating nodes deposit an amount of currency (deposit) and receive a reward if they have behaved honestly and, on the contrary, they are penalized if they do not, losing their deposit. The nodes bet on the blocks that will be added and if the block turns out to be correct, they receive the reward, i.e., betting on the consensus implies winning coins, while betting against the consensus implies losing them. This system of incentives and penalties maintains the consistency of the network.

- **Proof-of-Elapsed Time (PoET)**

Each participant requests a timeout from their local trusted enclave. The participant with the shortest timeout is next to propose a block, after waiting the allotted timeout. Each local trusted enclave signs the function and the result so that other participants can verify that no one has cheated on the timeout.

- **Proof-of-Space (Proof-of-Capacity)**

In this case, the user "pays" with hard disk space. The more hard-disk space the user has, the better is the chance of extracting the next block and earning the block reward. The algorithm generates large data sets known as "plots", which must be stored on the users' hard disk. The more plots the user has, the better is the chance of finding the next block on the chain. *Burstcoin* is the only cryptocurrency that currently uses a form of proof of capability.

- **Practical Byzantine Fault Tolerance (PBFT)**

This is a consensus algorithm that is normally used for consensus in distributed system but does not really meet the requirements for economic consensus on Blockchains since PBFT becomes infeasible in networks with a high number of nodes due to the required communication; Blockchain technologies using PBFT only rely on a reliable subnetwork of participants to establish consensus. Such a consensus algorithm is popular in private networks, being currently employed in *Hyperledger Fabric*, as it provides a way to reach consensus in a Blockchain where the majority of nodes are assumed to be "trusted" or non-malicious.

- **Istanbul Byzantine Fault Tolerance (IBFT)**

IBFT is a variant of the PoA algorithm. Moving away from the more technical aspects of IBFT, the most important fact is that it is, along with Raft[2], one of the consensus algorithms employable in *Quorum* networks. In the same way as PBFT, it makes sense mainly in private Blockchain deployments.

[2] Raft is a CFT consensus algorithm

5.2.3.2 Private vs public

Blockchains can be public or private:

- **Public**

A public Blockchain is open to the public and anyone can join without specific permission. All people who join the network can read, write, and participate in this network that is not controlled by anyone in particular.

- **Private**

Private Blockchains are based on invitation and anyone who wants to access it must ask for permission from the governing body of the Blockchain. They allow different levels of access that determine which users can write, read and audit the Blockchain. In this case, data is not public.

The main advantage of a private Blockchain is related to the control over the participants of the network, which is highly recommended in MEDINA, where the network should not be open to the public. In addition, the number of nodes needed to set up the network is limited, so the network is faster, more efficient and more convenient in terms of time and energy consumption. This is mainly because the consensus in public Blockchains is more complex since it is necessary to protect the network from untrusted nodes, so extra verifications and operations must take place, while in private Blockchains it does not happen because the nodes which are in the network are under control; it logically takes more time to synchronize a network and reach consensus when more nodes are involved in the consensus process. Finally, private Blockchains can be free of charge, which is highly recommended for the MEDINA audit trail system.

5.2.3.3 Technical comparison

This section presents some of the most well-known Blockchain technologies with their main characteristics.

- **Bitcoin**

Bitcoin [18] is a protocol conceived in 2008 by Satoshi Nakamoto, an anonymous person, that promises decentralized payments between parties with no central authority using peer-to-peer technology. Bitcoin promises to send value in form of tokens between different actors by paying a small amount of money as fee, offering the promise of lower transaction fees than traditional online payment mechanisms. There is no physical bitcoin, only balances kept on a public ledger that everyone has transparent access to.

All bitcoin transactions are verified by a massive amount of computing power, which is commonly known as mining. Regarding their Blockchain characteristics, Bitcoin is public and permission less, as anyone has access to the shared ledger and can participate in the network.

- **Ethereum**

Ethereum [19] includes open access to digital money and data-friendly services for everyone. It is a community-built technology behind its cryptocurrency ether (ETH). It also supports decentralized programmable Smart Contracts, which use ether to work. These Smart Contracts are implemented using Solidity language. Ethereum also has transaction fees, so users must pay a small amount of money to use the network, similarly to Bitcoin. It features a throughput of approximately 20 transactions per second, bettering Bitcoin. It is a public and permission less Blockchain.

- **Hyperledger Fabric**

Hyperledger Fabric [20] is a platform for the implementation of distributed solutions. It is based on Blockchain, so it can take advantage of all the benefits provided by this technology. Fabric implements Smart Contracts using Go as programming language.

Hyperledger Fabric, unlike Bitcoin and Ethereum, is private which means that permissions are required for third parties to access the network, and it is also permissioned, so it is possible to set different permissions to different nodes in the network. This marks a profound difference with Ethereum when it comes to forming consensus, since in Ethereum the roles and tasks required to reach consensus are identical. In addition, due to its nature, it allows the implementation of private channels, so that it is possible to share information only with certain parties. Unlike Bitcoin or Ethereum, it does not have mining or its own token, so it is not possible to give it cryptocurrency functionalities. In addition, due to the smaller size of the networks, it does not present as many scalability problems as the previous ones.

- **Quorum**

Quorum [21] is, according to the project page, an enterprise-focused version of Ethereum. The differences with Ethereum are therefore notable; on the one hand, it is permission-oriented and works on private networks. It also promises high speed and high performance, although logically it should not be compared with technologies such as Ethereum or Bitcoin, since, as they are focused on public networks, it is to be expected that performance and speed will be much lower due to a larger number of nodes.

Being based on Ethereum, it supports the use of Smart Contracts and has a token. Also, according to the project page, since it runs on Ethereum, it is easy to incorporate Ethereum functionalities into Quorum. As for the consensus algorithm, it uses PoS, although it can also work with other consensus algorithms.

- **Corda**

Corda [22] is an open-source project based on Blockchain technology and designed to be used mainly by financial institutions. In terms of scalability, it has the same particularities as Hyperledger Fabric, as well as the consensus mechanism. However, Corda uses what are known as notary nodes, which provide evidence that a transaction has been carried out. This way of reaching consensus is state-based.

Like Hyperledger Fabric, it is private and permission-oriented and implements Smart Contracts, which can be mainly implemented in Java or Kotlin. Like Hyperledger Fabric, it does not have its own token.

- **Hyperledger Sawtooth**

Hyperledger Sawtooth [23] is similar to Hyperledger Fabric, but in this case, it is designed to operate in IoT devices with little human interaction. It incorporates the consensus mechanism PoET. It has become well-known due to its ease of integration into security hardware solutions. In addition, it provides some advances over Hyperledger Fabric such as the ability to execute transactions in parallel and offers support for multiple languages and Ethereum. However, the project is still at a very early stage of development and, in addition, having been developed by Intel, there are doubts about the range of hardware devices that will be able to work with this system.

- **Hyperledger Besu**

Hyperledger Besu [24] is a java-based Ethereum client designed to be enterprise-friendly for both public and private permissioned network use cases. It can also be run on test networks such as *Rinkeby*, *Ropsten*, and *Görli*. Hyperledger Besu includes several consensus algorithms including PoW, and PoA (IBFT, IBFT 2.0, Etherhash, and Clique). Its comprehensive permissioning schemes are designed specifically for use in a consortium environment. The project, formerly known as Pantheon, joined the Hyperledger family in 2019, adding for the first time a public blockchain implementation to Hyperledger's suite of private blockchain frameworks. Whereas Hyperledger Fabric is a private protocol designed from the ground up to support enterprise-grade solutions, Besu seeks to utilize the public Ethereum network.

Regarding Hyperledger Besu, the Besu client is designed to be highly modular to ensure that key Blockchain features such as consensus algorithms can be easily implemented and upgraded. The goal here is to provide businesses with the means to easily configure Ethereum according to their needs while enabling smooth integration with other Hyperledger projects, such as Hyperledger Fabric.

Its smart approach of using the Ethereum Blockchain affords developers enough flexibility to build public or permissioned solutions based on the specific requirements of each use case. Rather than a comparison between Hyperledger Besu and Hyperledger Fabric, it is important to remark than both technologies are complementary and solve different problems. However, Hyperledger Besu presents advantages against Hyperledger Fabric in terms of interoperability, because it can be integrated as an enterprise client in any Ethereum network. It also has compatibility with Quorum. Hence, it fulfils more integration requirements than Hyperledger Fabric, which can only use its own network. Also, due to its compatibility with Ethereum, Hyperledger Besu allows the use of tokens.

- **Amazon QLDB**

Amazon Quantum Ledger Database (Amazon QLDB) [25] is a fully managed ledger database that provides a transparent, immutable, and cryptographically verifiable transaction log owned by a central trusted authority. Amazon QLDB can be used to track all application data changes and maintain a complete and verifiable history of changes over time. Amazon QLDB is a new class of database that helps eliminate the need to engage in the complex development effort of building your own ledger-like applications. With QLDB, the history of changes to your data is immutable.

Amazon QLDB works as a "Blockchain-as-a-Service" (BaaS) where Amazon provides the infrastructure. One of the main drawbacks is that governance is fully managed by Amazon and data is stored on Amazon´s side, centralizing the storage in a single provider. Also, it has associated costs as it works as-a-Service.

- **BigchainDB**

BigchainDB [26] was mainly developed to combine the best characteristics of the "traditional" distributed database and the "traditional" Blockchain. It uses MongoDB as database and allows queries over the stored data, while preserving the immutability and decentralization; and Tendermint [27] as Blockchain framework. It has low latency and presents a better throughput than other Blockchains, such as Bitcoin and Ethereum. However, this is not a fair comparison as it is permissioned and uses BFT algorithm to reach consensus, which is significantly faster than other algorithms used in public networks. One of its strong points is that it can be easily integrated in traditional stacks, providing a decentralized and immutable ledger where data and transactions can be stored.

- **ChainifyDB**

ChainifyDB [28] presents itself as a solution to integrate existing databases with Blockchain technology. It proposes the installation of a lightweight Blockchain layer on top.

ChainifyDB is a permissioned Blockchain layer which can be integrated into an existing heterogeneous database landscape adding a low overhead (8.5%) on the underlying database systems. It also promises up to 6x higher throughput than Hyperledger Fabric.

- **CovenantSQL**

CovenantSQL [29] is a BFT relational database built on a standard SQLite, powered by a decentralized query engine. Hence, it seems to work as a private and permissioned Blockchain. It is an open-source alternative of Amazon QLDB. It also achieves decentralization by using peer-to-peer technology and keeps the integrity of the data stored in it. At the current date, they are still working on the whitepaper.

- **FlureeDB**

FlureeDB [30] is an enterprise Blockchain-based database solution that combines Blockchain's security, immutability, decentralization and distributed ledger capabilities with a feature-rich graph-style database. It is composed by a database and a permissioned Blockchain. Regarding the ledger, it can be kept private among a consortium of entities or public for everyone.

FlureeDB deviates from other Blockchain technologies, such as Hyperledger Fabric or Ethereum, by focusing on queries and being optimized for read performance. Hence, it can be used as a complement to these technologies, rather than a direct rival, by for example storing transactions´ data.

- **HBasechainDB**

HBasechainDB [31] is a big data storage system for distributed computing based on Blockchain. It achieves immutability and decentralization thanks to Blockchain and uses a HBase database. An HBase database [32] is a column-oriented non-relational database management system that runs on top of Hadoop Distributed File System (HDFS) and is fault-tolerant. This database is not compatible with structured query languages, such as SQL, so it clearly deviates from other alternatives, such as CovenantSQL or ChainifyDB. Its scope seems therefore quite limited to big data applications, in particular those running Hadoop. Finally, HBasechainDB is permissioned, as only authorised nodes are able to submit transactions.

In practice, HBasechainDB follows a similar approach than BigchainDB, but it uses Hadoop database instead of MongoDB. However, HBasechainDB seems to be more appropriate for big data applications as it uses Hadoop database.

Although some of the more recent aforementioned technologies, such as HBasechainDB or BigchainDB, present some advantages in terms of performance, they also present some concerns in terms of governance because the network is under the control of an enterprise (for example, in AmazonQLDB). In addition, most of developers are currently more familiar with more traditional Blockchain technologies, such as Ethereum or Hyperledger Fabric, which also have a big community behind them. Hence, they are much more appropriate in terms of support and compatibility for the audit trail in MEDINA.

5.2.3.4 Blockchain technology for MEDINA

On the one hand, section 5.2.2 has concluded that Blockchain is a suitable technology for the MEDINA audit trail. On the other hand, section 5.2.3.2 has also concluded that a private Blockchain network is more suitable for the MEDINA audit trail. Taking these two ideas into consideration, and the analysis of Blockchain-related technologies from section 5.2.3.3, the technologies whose features better fit MEDINA audit trail requirements are: **Hyperledger Fabric** and **Quorum** (traditional general purpose private Blockchains).

Hyperledger Fabric aims to provide the basis for an extensible, modular, business-focused architecture that can be adopted by organizations in a variety of sectors. In contrast, Quorum is presented as an application-independent platform, with numerous differences and adaptations with respect to Ethereum but focusing on business needs. Therefore, although different in their initial approaches, both technologies aim to solve the problems associated with consortiums of professionals and organizations.

The following table presents a comparison between Hyperledger Fabric and Quorum, the most known private technologies.

Table 10. Overview of the most suitable Blockchain technologies features for MEDINA audit trail

Feature	Hyperledger Fabric	Quorum
Description	Modular Blockchain platform	Distributed registration protocol for enterprises and Smart Contracts platform.
Governance	Linux Foundation	J.P. Morgan (now, ConsenSys)
Operation mode	Permissioned (private)	Permissioned (private)
Participation	Per organization	Per node
Permission level	Fine grained (creation of users, deployment of Smart Contracts...)	Simple (validating node or not)
Message privacy	Yes	Yes
Type of privacy	By communications channel	By transaction
Private communications	Establishment at the beginning. Difficult to dissolve	Indicated in each message. No fixed link
Consensus	-SOLO (ordering) -Kafka (ordering) -Simplified BFT (future) -Practical BFT (future)	-Raft (no BFT) -Istambul BFT
License	GPL / LGPL	GPL / LGPL
Confirmation time	Instant	Instant
TPS	450-900 (theoretical)	800 (theoretical)

Feature	Hyperledger Fabric	Quorum
Transaction logs	Hash-linked blocks	Hash-linked blocks

As it can be deduced from the previous table, both technologies have similar features that can be useful. For that reason, and just taking simplicity in the network management into consideration, **Quorum has been considered as the Blockchain network technology for MEDINA audit trail**.

5.2.3.5 Future potential exploitation

Although the Blockchain network will be provided as a service by TECNALIA during the project-life for audit trail implementation validation purposes, possible exploitation plans for the system have been also analysed.

Nowadays, the European Commission is currently directly supporting the development of the Blockchain technology by means of several public initiatives following European Blockchain Strategy [33] as it has a high potential to transform Europe's industries and citizens' lives. One of these initiatives is the European Blockchain Services Infrastructure (EBSI) [34], a network of nodes distributed across Europe that will provide cross-border digital public services that comply with European regulation and meet the highest standards of security and privacy. The main objective of EBSI is to enable Blockchain technology to improve the way citizens, governments and businesses interact by means of becoming a network in which member states can use the existing infrastructure in a flexible way to cooperate across cross-border public services, connect existing solutions or integrate specific services.

MEDINA audit trail service perfectly fits the EBSI objectives, so, it could be considered a suitable Blockchain framework for future exploitation.

Note that the implementation of the trustworthiness system is included in WP3 [2].

6 MEDINA Automation of the Cloud Security Certification Life-Cycle

After evaluating the assessment results, i.e., aggregating and weighing them (see Section 4), a decision needs to be made about how these results should influence the state of the respective certificate. This management of certificates is done by the life-cycle manager. Note that in future iterations, the Risk Assessment (T4.4) will process the results of the certification evaluation before forwarding them to the life-cycle manager.

This section describes the MEDINA approach to managing the certificate life-cycle and consists of three parts. First, an analysis of the risks involved in managing the life-cycle of certificates automatically is conducted. The analysis concludes with a discussion of mitigative technologies, with a special focus on *smart contracts*. Second, the MEDINA implementation for a state machine that reflects the EUCS-defined certificate states is presented. Third, an approach to integrate *self-sovereign identities* into the life-cycle management is presented and discussed.

6.1 Risks and Mitigations in Certificate Management

Certificate management ensures that certificates reflect the current security level of a cloud service by translating evaluation results into a certificate state, and possibly making that state public. There are various risks that threaten this activity, and different possibilities to counter these risks.

6.1.1 Potential Risks

In traditional certification approaches, issued certificates are published and often can be verified with the certification authority. In this case a (potential) customer may, e.g., use the certification body's website to see if the auditee's name is listed there.

Reputation damage: One potential risk in certificate management concerns the auditee's reputation which can significantly be impacted by the evaluation results, which an automated certification process continuously generates. If, for instance, a component or data flow is manipulated to modify the outcome of the evaluation of assessment results, a competitor may damage a cloud service provider's reputation. At the same time, a malicious auditee may also try to manipulate the logic of this evaluation process to generate compliant results that ultimately result in the desired certificate state. The publication of a certificate's state — or state change — therefore needs to be protected from intentional and unintentional interference.

Denial of service: Also, certificate management needs to ensure that the current state of any certificate is available to be viewed (and verified) by stakeholders, e.g., in a public registry. If a certificate is not available, it is not possible to fully trust the claimed security of the respective auditee. Usually, however, the verification of a certificate is not time-critical, so a temporary non-availability of a certificate is neither very likely, nor is it very harmful.

Loss of trust: A further, more abstract, risk is the loss of trust that is put into the certification process and its actors. The certificate's value highly depends on that trust — an erroneous certificate state change could therefore also severely hurt the trust into the continuous certification process, and the certification, itself.

Summarizing, the protection goals that are relevant are the following.

- Confidentiality of evaluation details, such as non-compliances of specific resources (only the certificate state is public)
- Integrity of the certificate state
- Availability of the certificate

Attack vectors towards these goals and assets are therefore as follows:

1. **Modify the logic of the certificate management component:** a malicious attacker may try to modify the certificate manager to generate non-compliant results, e.g., to hurt competitors.
2. **Forge a certificate:** an attacker may try to create an illegitimate certificate that is trusted by potential customers.
3. **Delete a certificate:** an attacker may try to delete an existing certificate, e.g., to hurt a competitor.
4. **Deny the retrieval of a certificate:** using a denial-of-service attack, an attacker may try to prevent that the existence or state of the certificate can be retraced.
5. **Disclose sensitive certificate details:** an attacker may disclose details about the state of a certificate, e.g., non-compliance details of a suspended certificate, possibly revealing vulnerabilities of the CSP.

As described above, the impacts can include reputation damage to the auditee, but also reputation damage to the certification process, the certificate, and the certification authority.

6.1.2 Discussion of Possible Mitigations

Note that in this version of the deliverable, this discussion focuses on the possibility of using smart contracts to mitigate the threats identified above. Future versions will discuss other approaches in more detail as well.

6.1.2.1 Smart Contracts

One possibility to protect the integrity of the certification management logic (attack vector 1) is to use *smart contracts*. Ante [35] defines smart contracts as "*decentrally anchored scripts on blockchains or similar infrastructures that allow the transparent execution of predefined processes*". Historically, the term *smart contract* has not necessarily been associated with blockchains. For example, Röscheisen et al. [36] described a smart contract already in 1998 as a "*digital representation of an agreement between two or more parties" that has "a structured […] interface, code that implements behavior, state (e.g. the validity status, the number of times a right was exercised, etc.), and a set of textual descriptions*".

In the documentation of Ethereum, the most popular platform for the deployment of blockchain-based smart contracts, a smart contract is defined as "*a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain*"[3]. Most cryptocurrencies, e.g., Bitcoin, use a blockchain to store transactions between accounts. To make the execution of smart contracts possible, Ethereum also stores code and data on the blockchain to enable the execution of the Ethereum Virtual Machine.

It is furthermore important to note that there is a difference between a smart contract and a *legal* contract: a smart contract, e.g., anchored on a public blockchain, does not necessarily represent a legally binding document[4].

The goal of using smart contracts is usually the elimination of trusted third parties. One reason is that trusted third parties may sometimes not be fully trusted by all stakeholders. Also, they incur additional cost and overhead into a transaction. For example, certification audits may consume many person days to prepare documentation, conduct interviews, create reports, etc. The most prominent examples of avoiding trusted third parties are cryptocurrencies, like Bitcoin

[3] https://ethereum.org/en/developers/docs/smart-contracts/

[4] Note that the authors are no legal experts, but analyse the possibility of using smart contracts merely from a technical perspective.

and Ethereum, which aim at eliminating the need for financial institutions to manage a currency and accounts.

In the traditional certification process, trust is mainly established via the trusted certification authority – a reputable third party that has no interest in issuing an undeserved certificate. In the continuous process, in contrast, trust needs to be established through a reliable design and technical implementation that guarantee the correct management of certificates.

In the following, some inherent risks of using smart contracts are described.

- A risk of using smart contracts is that they could be deployed including bugs and vulnerabilities, which may only be discovered after their deployment. While various approaches have been proposed to validate a smart contract's purpose and to eliminate bugs before their deployment, this risk can never be fully eliminated.
- Also, the environmental impact of blockchain technologies should be considered. Storing a large number of transactions can, depending on the algorithm, result in high amounts of energy consumption.
- A further considerable disadvantage of smart contracts is that there is no possibility for remediation or consideration if the contract fails. Traditional contracts often include a severability clause which may define that the purpose of the contract is still effective even though a part of it emerges to be unenforceable. This way, the general purpose of the contract can be upheld. In smart contracts, in contrast, there is no room for interpretation or consideration. In the context of certification, this means that in case a part of the contract becomes, e.g., outdated, illegal, or unenforceable, there is no possibility to change its scope or logic.

The topic of using smart contracts for different purposes has also been discussed in the literature. Some works have investigated, for example, how to transform business processes to smart contract-based processes that eliminate intermediaries and work more efficiently, e.g., proposing frameworks [37] and compilation processes [38].

Few works actually investigate the challenges that occur and that have to be solved to port business processes to the blockchain. For example, Carminati et al. [39] identify challenges for the application of smart contracts for inter-organizational business processes. As such, their results are largely applicable to cloud certification as well, which is a business process between several organizations, possibly including a CAB, an auditee, and one or more cloud vendors.

They identify five challenges which are discussed in the context of certification in the following:

- **Data integrity:** Important data that is processed by smart contracts has to be integrity-protected as well. Smart contracts should therefore store all relevant data in protected transactions. In the context of certification, this challenge also raises the question of input integrity. For example, a smart contract may obtain input data from a cloud service, e.g., about encryption configurations. If these data, however, are maliciously modified then they will be used in the unalterable logic of the smart contract which in turn produces outcomes that are stored unchangeably on the blockchain.
- **Data confidentiality:** While it is a current research problem to allow for confidential blockchain transactions, it is not a standard feature. The data that a smart contract generates and uses are therefore public – assuming that a public blockchain is used. When making certification decisions, this public information could potentially reveal sensitive information about the CSP's security problems.
- **Confidentiality of the process:** Carminati et al. [39] also raise the issue of confidentiality of smart contracts themselves, i.e., their program logic, since the process flow itself may

reveal sensitive information. When implementing certification processes, this is not a relevant issue, since the workflow of a certification process, as well as its decision criteria, are usually defined in public documents.

- **Trust in the correct execution of the process:** Process trust has to be established for all participants in the business process. One threat to this trust is the possible data breaches and tampering attacks that may happen when the smart contract interacts with off-chain components which are not integrity-protected. This is also a major issue for implementing certificates as smart contracts, since the certification process itself requires trust by customers in this process. In traditional certification approaches, this trust is established through the auditors who represent a trusted third party.
- **Data provenance:** Data provenance refers to the origin of data and its "history". In MEDINA, there is an inherent trust assumption for the tools that gather and assess evidence, so this data's provenance is assumed to be verified. In future work, however, the issue of making the provenance of evidence that is, e.g., gathered from a public cloud provider, should be addressed.

In summary, smart contracts can be used to reliably execute a piece of code, e.g., for translating evaluation results into a certificate state according to pre-defined criteria. Yet, elements of the certification pipeline, i.e., all systems and tools that contribute to the continuous certification including evidence gathering, evaluation, and certificate management, that are not (or cannot be) deployed in an integrity-protected environment, such as a Blockchain, can severely limit the usefulness of a blockchain-deployed smart contract. For example, APIs for the gathering of evidence may change, configurations for the smart contract may change (e.g., the service location or scope), or the requirements for publishing or managing the requirements may change (e.g., the states and their conditions). Also, the data transmission from off-chain elements to the smart contract may be attacked. As soon as such a condition changes, the smart contract may become non-functional and the continuous certification process may be interrupted.

The question therefore is whether these conditions can be assumed to remain unchanged, and whether a smart contract can mitigate the previously identified risks, e.g., the risk of malicious modification of the certificate management logic. On the one hand, smart contracts can reliably protect the integrity and execution of a piece of code. They can therefore be seen as a mitigation for attack vectors 1 and 2 (see Section 6.1.1). On the other hand, this mitigation introduces new risks, e.g., unfixable bugs, disclosure of sensitive information, and the challenge of protecting the integrity of the other parts of the certification pipeline remain.

6.1.2.2 Verifiable Credentials

The current conformities attestation procedure is based on physically signed reports issued after an auditing process. The aim of MEDINA is to provide a useful tool for making this process easier. In this sense, verifiable credentials appear as a suitable way to secure the authenticity of digitalized reports emitted by trusted authorities.

In the physical world, a credential might consist of information associated to specific attributes or properties being asserted by the issuing authority about a subject (for example, nationality, classes of vehicle entitled to drive, date of birth, etc.). In the digital world, a digital credential can also represent the same information that a physical credential represents; it is a way to digitalize physical credentials.

Furthermore, the addition of more advanced technologies, such as digital signatures, makes credentials more tamper-evident and more trustworthy than their physical counterparts, converting them into verifiable credentials.

These verifiable credentials are currently considered as part of an actor identity. That is why they are considered as the main identity components behind the Self Sovereign Identity approach. The following section describes how an SSI approach can be useful for MEDINA.

6.2 Self-Sovereign Identity in Certification

6.2.1 Self-Sovereign Identity (SSI) Concept

Self-Sovereign Identity (SSI) is a kind of digital identity which provides control, security and portability. The individual (or organisation) to whom the identity belongs, owns, controls and manages their identity completely, i.e., the individual is their own identity provider, there is no third party that can claim to "provide" the identity and no one can take away the user's self-sovereign identity.

Christopher Allen set out the 10 principles that a system implementing SSI must have [40]:

- **Existence:** Users must have an independent existence. Any self-sovereign identity is based on the “I” which is the heart of identity. It cannot exist only digitally. SSI simply makes public and accessible certain aspects of the "I" that already exists.
- **Control:** Users must control their identities. It is necessary to have algorithms that the subject can understand and that in turn ensure the continued validity of their identity. They must be able to refer to it, update it and even hide it. This does not mean that the subject controls all assertions about itself; other subjects or entities may make assertions about the subject but should not control the identity of the subject.
- **Consent:** Users must agree to the use of their identity. Any identity system is based on sharing that identity and its notifications. However, data sharing should only occur with the consent of the user. Although other users may make assertions about him, the user must offer consent for it to be shared.
- **Protection:** The rights of the user must be protected. When there is a conflict between the need for identity and the rights of the user, the network must act in the interest of preserving the user's freedom and rights rather than the need of the network. To ensure this, identity authentication must occur through independent algorithms that are censorship-resistant, force-resistant and run in a decentralised manner.
- **Persistence:** Identities should be long-lived. Preferably, identities should last forever, or as long as the user desires. Although keys or data may need to be changed, the identity should remain. This point should not contradict "the right to be forgotten"; a user should have the right and the possibility to get rid of an identity if he or she wishes to do so, and statements about it should be modified or deleted accordingly.
- **Minimisation:** Disclosure of claims should be minimised. When disclosing data, this disclosure should involve the minimum amount of data necessary to perform the task. For example, if only a minimum age is requested, then the exact age should not be disclosed.
- **Interoperability:** Identities should be as usable as possible. Identities are of little value if they only work in limited niches. Nowadays, the goal is to make identity information widely available, crossing international borders to create global identities, without losing user control.
- **Transparency:** Systems and algorithms must be transparent. The systems used to manage and operate identities must be open, so that it is known how they work, how they are used and how they are updated. Algorithms should be free, open-source, well-known, and as independent as possible of any particular architecture.
- **Access:** Users must have access to their data. A user should always be able to easily access all assertions and data within their identity. This does not mean that the user can modify the assertions associated with their identity, but it does mean that they should

be aware of them. It also does not mean that users have equal access to each other's data, only to their own.

- **Portability:** Identity-related information and services should be portable. Identities should not be in the hands of a particular third party, even if it is a trusted entity that is expected to operate in the best interest of the user. The problem is that entities can disappear, and on the Internet, most do. Regimes may change, and users may move to different jurisdictions. Transportable identities ensure that the user remains in control of his or her identity and can also improve the persistence of an identity over time.

SSI is a digital identification scheme in which the subjects whose identity is created acquire responsibility for managing how, when and with whom they share their personal data. To this end, it allows the creation of "digital identity proofs" (presentations) based on his own identity attributes.

In general, there are three actors involved in the SSI schema:

- **Issuer:** provides verifiable credentials with identity attributes related to the user. It creates and signs credentials.
- **Owner:** locally stores and controls the credentials about oneself.
- **Verifier:** needs to identify a user's attribute or a set of them based on verifiable credentials by trusted issuers. Verifier does not need to store any user (owner) data, but only needs to verify it. This verification is based on validating the owner's provided credentials proof, in where requested claims by verifier are attested. Depending on verifier requirements this attestation could disclose credential claims values or be a private attestation based on ZKP (Zero Knowledge Proofs).

Regarding the information components involved in an SSI solution:

- **Credential:** it is a digital certificate containing identity attributes of the owner it is associated with. It is issued by the issuer.
- **Presentation**: it is a digital proof shared by the owner with the verifier to prove certain characteristics of the subject's identity based on the received credentials.

The verifier does not necessarily be related to the issuer, so, the only way to digitally prove credentials have been really issued by a trusted issuer, and have not been modified in any way, is by means of digital signatures. Digital signatures are based on applying a private key in a digital signature algorithm over a specific information. The signature can be verified using the same digital signature algorithm but with the associated public key. In the SSI context, digital signing is at least applied by the issuer to the credentials, becoming **Verifiable Credentials**, and by the owner to the presentations, becoming **Verifiable Presentations**. Public keys associated to issuers and holders should then be known. They could be saved in a centralized database, but there may be integrity issues due to a third-party being able to modify the public keys, availability issues (SPoF), due to probably system bringing downs and in addition the operator of this centralized database would have visibility of al relationships between different subjects.

For these reasons, instead of a centralized database, decentralized solutions have been widely considered for public key storage in SSI deployments. re:claim [41], for example, uses a distributed database like the GNU name system, while IPFS includes a specific method for public keys storage [42]. However, most of the current SSI implementations are backboned by Blockchain (or general Distributed Ledger Technologies, DLT), which can act as a suitable global repository for public key identifiers in SSI, as it solves several problems from traditional databases:

- **Trust**: Blockchain is based on a decentralized network of computers which is not owned by one single party, so it is not necessary to trust on any specific party.
- **Integrity:** Immutability is an inherent property of Blockchain guaranteeing tamper-proofed data.
- **Availability:** Blockchain is a network of computers across the globe and bringing down it is near to impossible.

Blockchain creates globally distributed databases that can serve as a source of truth for public keys without being subject to SPoF. This is the reason why Blockchain generally fits into the SSI infrastructure for registering and resolving public keys. In this sense, SSI uses Decentralized Identifiers (DID) as a unique and global identifier of every actor involved in the process. Each DID is associated to a DID Document, describing its properties, such as the associated public key (and additional public keys that are authorized to perform actions in its name) or service endpoints for interacting with the specific DID. It is necessary for the signing process in Verifiable Credentials by the issuer and in Verifiable Presentations by the owner. This architecture eliminates unnecessary third-party identity providers and highly reduces security risks. Therefore, the basic architecture of the Blockchain-based SSI solution is shown in Figure 6.

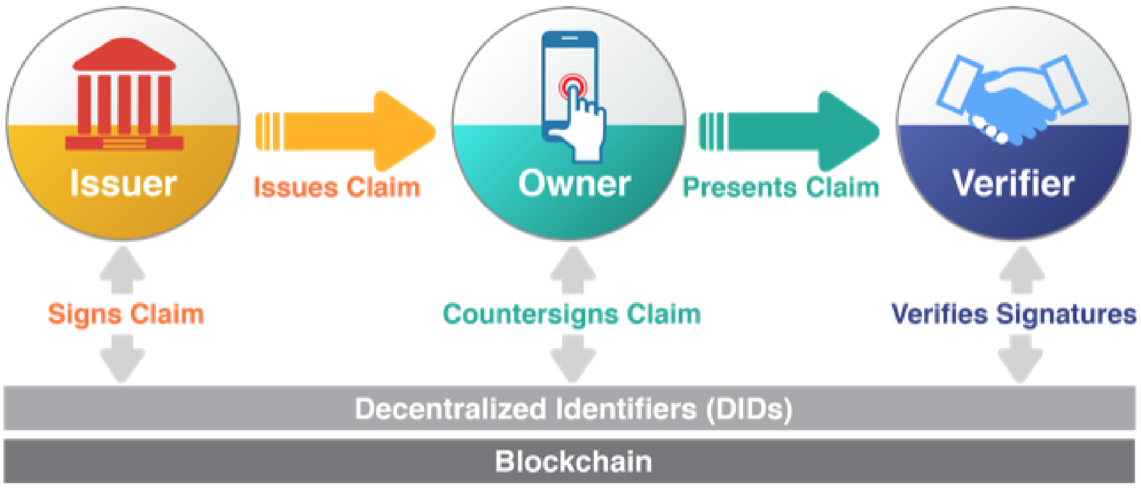

Figure 6. Overview of an SSI architecture using Blockchain

6.2.2 How to consider SSI in MEDINA

This section describes the way MEDINA will consider SSI based concepts, identifying involved actors and required interactions.

Nowadays, the legal entity that performs a conformity assessment of the cloud service providers against relevant regulations and standards is the Conformity Assessment Body (CAB). It carries out audit following the required regulations. Once the audits are completed, a conformity assessment results report is submitted, clearly identifying the detected non-conformities. In the cases where significant non-conformities are present, a new audit will be scheduled to verify the elimination of the detected non-conformities. This process is currently done in a manual way by means of arranging a meeting with the client service provider for reviewing and signing the report.

MEDINA, as a framework for helping in the audit processes, will also provide an additional tool for digitalizing the conformity assessment results report based on the information gathered by MEDINA framework. For this purpose, SSI concept will be considered, following the mappings presented in Figure 7.

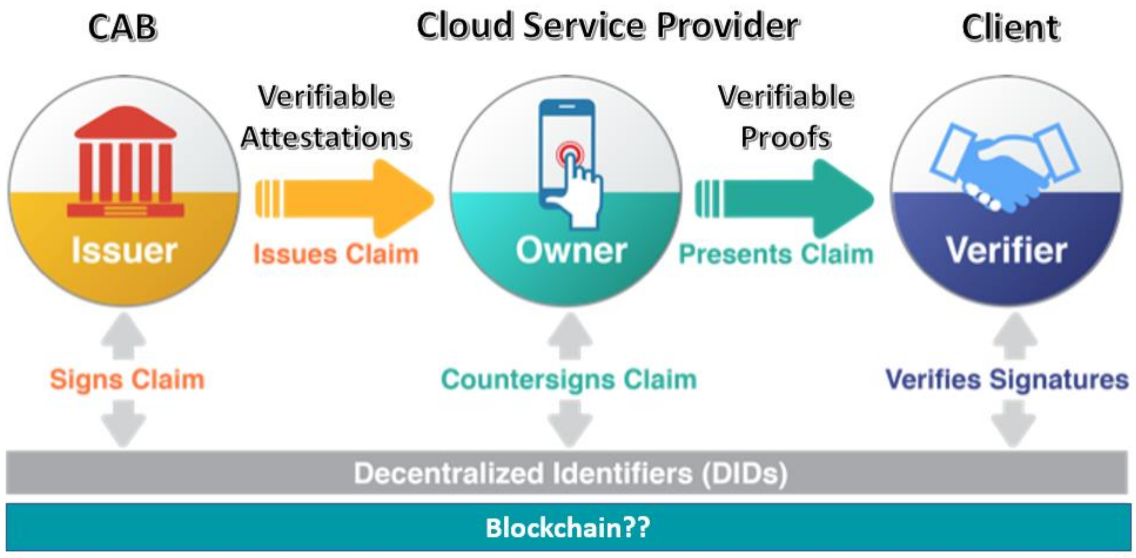

Figure 7. SSI applied to the MEDINA context

The main actors involved in this process related to MEDINA are:

- The **CAB** as the conformity assessment result report issuer (issuer in SSI). Its role will be based on adapting the information obtained from the MEDINA framework to a common data model defined for the conformity assessment result report and issuing verifiable credentials containing these details.
  The verifiable credentials could include "public" attestations for being save in a public registry (including, for example, the final conformity assessment result) as well as "private" or "confidential" attestations, for being privately saved with the cloud service provider (including, for example, the specific non-conformities). The CAB will be in charge of signing the different verifiable credentials as trusted authorities for this process.
- The **Cloud Service Provider** as the verifiable credential owner (owner in SSI). Its role will be to own, save and control the verifiable credential sharing as it is referred to itself. In addition, they will generate verifiable proofs from the public or private verifiable credentials issued by the CAB.
- In addition, and to make the mapping from MEDNA to SSI complete, **the cloud service provider clients** could act as verifiers. They could ask their cloud service providers for proofs of conformity assessments, being able to verify their validity.

Blockchain will be the Verifiable Data Registry, in which issuer and holder identification (DID, Decentralized Identifiers) will be saved in order to be able to verify signatures from Verifiable Credentials (from the issuer) and Verifiable Proofs (from the owner).

6.2.3 High-level Architecture

As this tool is additional to the base MEDINA framework, it will be completely provided as a proof-of-concept, just to validate the suitability of using SSI for future helping of the CAB operation. The specific deployment of the different services will be completely on TECNALIA premises, allowing the required interaction with the MEDINA framework.

Figure 8 shows the SSI based verifiable cloud security certification architecture showing its main components. The different services for the different actors have been identified as well as their relations.

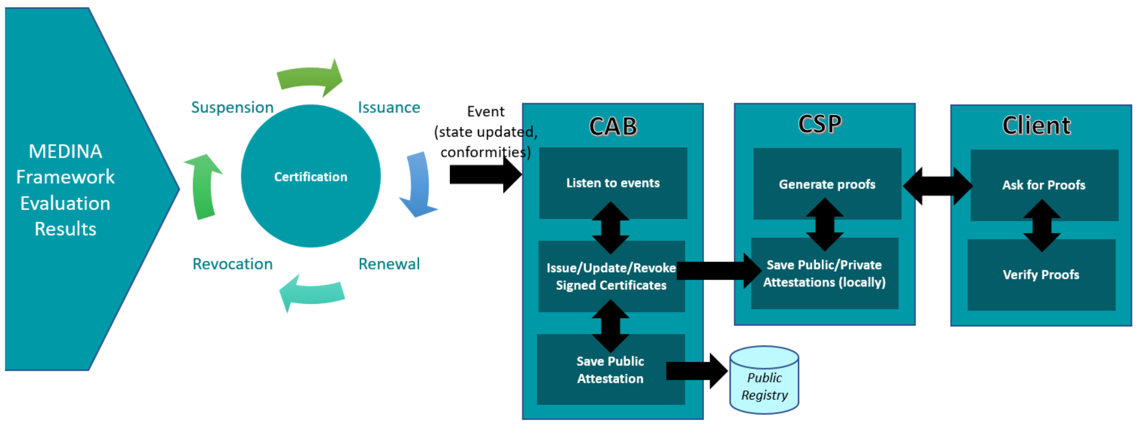

Figure 8.MEDINA SSI based verifiable cloud security certification architecture

The SSI based verifiable cloud security certification will receive input from the MEDINA framework. This input may include: the current certificate state, the conformities and/or non-conformities detected, etc. This information may be provided by means of events.

The CAB will need the following services:

- A service for listening to the events from the MEDINA framework.
- A service for issuing, updating and/or revoking the public and private attestations (verifiable credentials) about the CSP based on the input from the MEDINA framework. This service will adapt the format to the specific data model of the attestations.
- A service for automatically saving the public attestations in a public registry.

The CSP will need the following services:

- A service for receiving public and private attestations (verifiable credentials) from the CAB and locally saving them.
- A service for generating verifiable proofs to share with their clients based on the verifiable credentials from the CAB.

The CSP clients will need the following services.

- A service for asking the CSP for specific proofs (both associated to private or public attestations).
- A service for verifying signatures from the verifiable proofs.

More details will be provided in the following iterations of this deliverable.

6.3 Design and Implementation

Due to the potential risks of encoding certificates in smart contracts, the first iteration-implementation of the life-cycle manager component is independent from smart contracts and blockchains in general. In future iterations, the implementation will be integrated with the appropriate protection measures, e.g., the SSI concept presented above. The first integrated version of the Life-Cycle Manager is published as an open-source project[5].

[5] https://git.code.tecnalia.com/medina/public/life-cycle-manager

6.3.1 Functional Requirements

The following requirements, fully defined in D5.1 [4], are addressed (partly) in the first iteration:

- **ACLM.01:** Based on the quality evaluation results, the system will push appropriate entities (CAB) to issue and sign security certifications for the cloud service providers.
- **ACLM.**02: Based on the quality evaluation results, the system will push appropriate entities (CAB) to update the security certifications for the cloud service providers.
- **ACLM.**03: Based on quality evaluation results, the system will push appropriate entities (CAB) to revoke the security certifications for the cloud service providers.
- **ACLM.04:** The certificate life-cycle management component must continuously, i.e., in high-frequency intervals, convert the evaluation results from the CCE to the corresponding certificate state.
- **ACLM.06:** The certificate life-cycle management component must map the certificate states and assurance levels defined in the EUCS scheme.
- **ACLM.07:** The life-cycle management component must provide an interface for publishing the certificate status in a public registry by the corresponding entities (CAB).

Regarding **ACLM.08** (*The life-cycle management component can be implemented in a smart contract to ensure a tamper-proof execution*), a first evaluation of using smart contracts to implement certificates is included in this deliverable.

6.3.2 Certification Life-Cycle Phases

The EUCS defines the following certificate states:

- **New Certificate** for newly issued certificates, following an assessment with positive outcome.
- **Continued** for certificates that have been reassessed and should not reflect any changes.
- **Renewed** for certificates that have been reassessed and whose validity is extended. Updates to the certificate’s information may be added.
- **Updated** for certificates that have been reassessed and which remain valid, but need updates in its information.
- **Suspended** for certificates that have been reassessed with the outcome that the service does not conform to the requirements of the targeted assurance level anymore. The state is also entered if a periodic reassessment has not been conducted in due time.
- **Withdrawn** for certificates that have not been maintained after the suspension.

These states and transitions are reflected in the state machine model as shown in Figure 9. The dashed lines refer to the renewal flow where a certificate is first withdrawn and then renewed, e.g., to reflect a different assurance level.

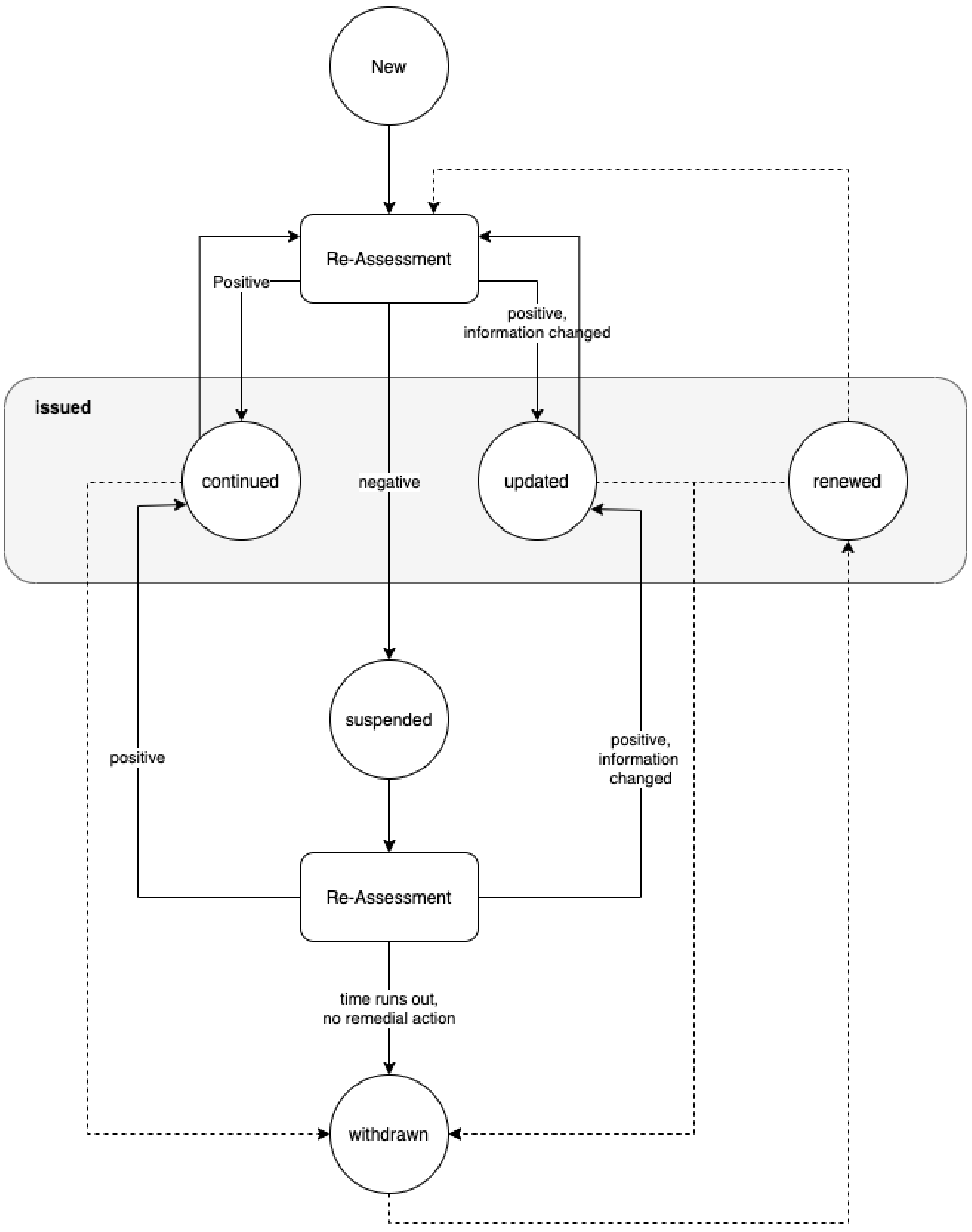

Figure 9. A state machine model of the EUCS phases

6.3.3 Implementation

The current implementation of the life-cycle manager (LCM) represents the state machine described in Section 6.3.2. It is written in Go and uses another open-source project, called *transition*[6] to implement the state machine logic. The LCM implementation is structured as follows.

[6] https://github.com/nowdo-hq/transition

- The main method in the *main/lcm* package initializes the database and the certificate state machine.
- The database logic is implemented in the *db* package. It uses either an in-memory database or a PostgreSQL database, which can be configured using an environment variable.
- Next, the *models* package contains all data models, currently a user model and a certificate model.
- The *cert* package contains the actual state machine logic: It creates a certificate state machine, defines its states, possible events, and state transitions.
- Finally, the *rest* package represents the API of the life-cycle manager. It listens to different HTTP routes for different commands, e.g., create a new certificate, or handle a deviation.

6.3.4 Interface for a Public Registry

After a decision about the certificate state has been made, it may be published/updated in a public registry.

To represent certificates, the ISO/IEC 17021 standard “Requirements for bodies providing audit and certification of management systems” [43] is used, which describes the information a certificate should contain. These include the following (simplified) data:

- Name and geographical location of the certified client
- Effective date of granting or changing the certification
- Expiry date or recertification due date
- Unique identification code
- Standard and/or other normative document, including issue status
- Scope of certification, e.g., type of activities, products and services
- Name, address and certification mark of the certification body
- Other information required by the standard and/or another normative document
- In the event of issuing any revised certification documents, a means to distinguish the revised documents from any prior obsolete documents.

This data model is foreseen for the life-cycle manager implementation and will be used to define an API for the publication of certificates.

6.4 Summary, Limitations and Future Work

The management of certificates is subject to considerable risks, including possible malicious attacks and technology and process shortcomings. Therefore, Section 6 has firstly presented an evaluation of technology options for managing certificates securely. Secondly, a concept for a self-sovereign identities concept has been described that can mitigate some of the identified risks. Thirdly, it has presented a first iteration of a state machine for certificates that will be integrated with the other components and improved with appropriate technologies to create a secure and trustworthy management of certificates.

The limitations the life-cycle manager faces lie most importantly in the potential for *meaningful* automation: For example, applying metrics like a risk value for the derivation of a certificate state neglects much of the information that is usually considered in a manual audit. The next deliverable (D4.2) will therefore explore in more detail to what degree the issuance and maintenance of a certificate should be automated.

Future work will also consider the integration of life-cycle manager with the self-sovereign identity system, the integration with a public registry, the integration with the risk assessment component, as well as the provision of a user interface.

7 Conclusions

The continuous certification of security properties in cloud services poses various challenges, including the continuous aggregation and evaluation of evidence, as well as the continuous management of certificates. Also, the protection of evidence integrity, i.e., its trustworthiness, is a major challenge, since it is essential to establish trust in the whole certification process.

This deliverable has introduced the first iteration of concepts and prototypes for a Continuous Evaluation component, an automated life-cycle manager, as well as the concept for the trustworthiness of evidence and assessment results (whose implementation is described in WP3).

The technology evaluations in this deliverable have shown that some emerging technologies, like blockchain and smart contracts, can provide benefits for the automation and protection of certificates. At the same time, they can introduce considerable overhead and new risks. Future work therefore has to carefully balance practical considerations of CSPs with appropriate security and automation measures.

Currently, the evaluation component is agnostic to the service's context, e.g., to the fact that some security requirements and some resources may be more relevant than others for a CSP's security posture. In future work, the evaluation of assessment results needs to be extended with a qualitative evaluation that considers such factors. Also, the current evaluations of distributed ledger technologies and smart contracts will be extended to paint a more comprehensive picture of how those technologies can contribute to protecting the integrity of automated audit processes and artefacts. Further future improvements include the review of more related work, improvements to the prototype implementations and their integrations into the overall MEDINA framework, and the further development of the architecture and data model.

References

[1] ENISA, “EUCS – Cloud Services Scheme,” [Online]. Available: https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme.

[2] MEDINA Consortium;, “D3.1-Tools and techniques for the management of trustworthy evidence-v1,” 2021.

[3] MEDINA Consortium, “D3.4-Tools and techniques for collecting evidence of technical and organisational measures-v1,” 2021.

[4] MEDINA Consortium, “D5.1-MEDINA Requirements, Detailed architecture, DevOps infrastructure and CI/CD and verification strategy-v1,” 2021.

[5] MEDINA Consortium, “D4.4 Methodology and tools for risk-based assessment and security control reconfiguration-v1,” 2021.

[6] MEDINA Consortium, “D4.5 Methodology and tools for risk-based assessment and security control reconfiguration-v2,” 2023.

[7] MEDINA Consortium, “D2.1 Continuously certifiable technical and organizational measures and catalogue of cloud security metrics-v1,” 2021.

[8] J. Luna, A. Taha, R. Trapero and N. Suri, "Quantitative Reasoning about Cloud Security Using Service Level Agreements," *IEEE Transactions on Cloud Computing,* vol. 5, no. 3, pp. 457-471, 2017.

[9] J. Luna, R. Langenberg and N. Suri, “Benchmarking cloud security level agreements using quantitative policy trees,” in *CCSW '12: Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, Raleigh North Carolina USA, 2012.

[10] A. Taha, R. Trapero, J. Luna and N. Suri, "AHP-Based Quantitative Approach for Assessing and Comparing Cloud Security," in *IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, 2014.

[11] J. Modic, R. Trapero, A. Taha, J. Luna, M. Stopar and N. Suri, “Novel efficient techniques for real-time cloud security assessment,” *Computers & Security,* vol. 62, 2016.

[12] EU FP7 SPECS, “Secure Provisioning of Cloud Services based on SLA management,” [Online]. Available: https://cordis.europa.eu/project/id/610795.

[13] S. Maroc and J. Biao Zhang, "Towards Security Effectiveness Evaluation for Cloud Services Selection following a Risk-Driven Approach," *International Journal of Advanced Computer Science and Applications (IJACSA),* vol. 11, no. 1, 2020.

[14] I. Kunz and P. Stephanow, "A process model to support continuous certification of cloud services," in *IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, 2017.

[15] M. Anisetti, C. A. Ardagna, E. Damiani and F. Gaudenzi, "A semi-automatic and trustworthy scheme for continuous cloud service certification," *IEEE Transactions on Services Computing,* vol. 13, no. 1, pp. 30--43, 2017.

[16] P. Torr, “Demystifying the threat modeling process,” in *IEEE Security & Privacy*, 2005.

[17] R. M. Blank, "Guide for conducting risk assessments," in *Citeseer*, 2011.

[18] S. Nakamoto, "Bitcoin. A peer-to-peer electronic cash system.," *Decentralized Business Review,* no. 21260, 2008.

[19] V. Buterin, "Ethereum white paper. GitHub repository," 2013. [Online]. Available: https://ethereum.org/en/whitepaper/. [Accessed September 2021].

[20] “Hyperledger Fabric,” [Online]. Available: https://www.hyperledger.org/use/fabric. [Accessed September 2021].

[21] Consensys, “Quorum,” [Online]. Available: https://github.com/ConsenSys/quorum. [Accessed September 2021].

[22] M. Hearn, "Corda: A distributed ledger. Corda Technical White Paper," 2016.

[23] H. Sawtooth. [Online]. Available: https://www.hyperledger.org/use/sawtooth. [Accessed September 2021].

[24] "Hyperledger Besu explained," [Online]. Available: https://limechain.tech/blog/hyperledger-besu-explained/. [Accessed September 2021].

[25] “What is Amazon QLDB?,” [Online]. Available: https://docs.aws.amazon.com/qldb/latest/developerguide/what-is.html. [Accessed September 2021].

[26] “About BigchainDB,” [Online]. Available: https://www.bigchaindb.com/. [Accessed September 2021].

[27] "Tendermint," [Online]. Available: https://tendermint.com/. [Accessed September 2021].

[28] F. M. Schuhknecht, A. Sharma, J. Dittrich, Agrawal and Divya, "chainifyDB: How to get rid of your Blockchain and use your DBMS instead," *CIDR,* 2021.

[29] “CovenantSQL-The Blockchain SQL Database,” [Online]. Available: https://covenantsql.io/. [Accessed September 2021].

[30] “Fluree – The Web3 Data Platform,” [Online]. Available: https://flur.ee/. [Accessed September 2021].

[31] M. S. Sahoo and P. K. Baruah, "Hbasechaindb–a scalable blockchain framework on hadoop ecosystem," in *Asian Conference on Supercomputing Frontiers*, 2018.

[32] “What is HBase?,” [Online]. Available: https://www.ibm.com/topics/hbase. [Accessed September 2021].

[33] European Commission, “Blockchain Strategy,” [Online]. Available: https://digital-strategy.ec.europa.eu/en/policies/blockchain-strategy. [Accessed September 2021].

[34] European Commission, “European Blockchain Services Infrastructure,” [Online]. Available: https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/EBSI. [Accessed September 2021].

[35] L. Ante, "Smart contracts on the blockchain–a bibliometric analysis and review," *Telematics and Informatics,* 2020.

[36] M. Röscheisen, M. Baldonado, K. Chang, L. Gravano, S. Ketchpel and A. Paepcke, “The Stanford InfoBus and its service layers: Augmenting the Internet with higher-level information management protocols,” *Digital Libraries in Computer Science: The MeDoc Approach,* pp. 213--230, 1998.

[37] W. Viriyasitavat, L. Da Xu, Z. Bi and A. Sapsomboon, "Blockchain-based business process management (BPM) framework for service composition in industry 4.0," *Journal of Intelligent Manufacturing,* vol. 31, no. 7, pp. 1737--1748, 2020.

[38] L. García-Bañuelos, A. Ponomarev, M. Dumas and I. Weber, "Optimized execution of business processes on blockchain," in *International conference on business process management*, 2017.

[39] B. Carminati, E. Ferrari and C. Rondanini, "Blockchain as a platform for secure inter-organizational business processes," in *Carminati, Barbara, Elena Ferrari, and Christian Rondanini. "Blockchain as a platform for secure inter-organizational business processes." 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC),* 2018.

[40] C. Allen, “Self-Sovereign Identity Principles 1.0,” [Online]. Available: https://github.com/WebOfTrustInfo/self-sovereign-identity/blob/master/self-sovereign-identity-principles.md. [Accessed September 2021].

[41] M. Schanzenbach, G. Bramm and J. Schütte, "reclaimID: Secure, self-sovereign identities using name systems and attribute-based encryption," in *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018.

[42] "IPID DID Method," December 2018. [Online]. Available: https://did-ipid.github.io/ipid-did-method/. [Accessed October 2021].

[43] "ISO/IEC 17021 Conformity assessment — Requirements for bodies providing audit and certification of management systems," International Organization for Standardization (ISO), 2015.