D2.5 Report on security and trust management for CECC

Document Summary Information

Project Identifier	HORIZON-CL4-2022-DATA-01. Project 101093129		
Project name	Agile and Cognitive Cloud-edge Continuum management		
Acronym	AC^3		
Start Date	January 1, 2023	**End Date**	December 31, 2025
Project URL	www.ac3-project.eu		
Deliverable	D2.5 Security and trust management for CECC		
Work Package	WP2		
Contractual due date	M14: 29th February 2023	**Actual submission date**	
Type	R- Document, report	**Dissemination Level**	PU – Public
Lead Beneficiary	EUR		
Responsible Author	Sofiane MESSAOUDI (EUR)		
Contributors	Adlen Ksentini, Ayoub Mokhtari, Sofiane MESSAOUDI, Abd Elghani MELIANI (EUR); John Beredimas, Athanasios Kordelas (CSG); Souvik Sengupta, Ali Nikoukar (IONOS)		
Peer reviewer(s)	Vrettos MOULOS (UPR) & Elias DRITSAS (ISI)		

Funded by the European Union

AC^3 project has received funding from European Union's Horizon Europe research and innovation programme under Grand Agreement No 101093129.

Revision history (including peer reviewing & quality control)

Version	Issue Date	% Complete	Changes	Contributor(s)
v1.0	10/11/2023	5%	Initial Deliverable Structure (ToC)	Sofiane MESSAOUDI (EUR)
V1.1	30/11/2023	45%	Security of Data Management	Souvik Sengupta (ION), Dimitrios Amaxilatis (SPA)
V1.2	15/12/2023	65%	API Security section filled	John Beredimas (CSG), Athanasios Kordelas (CSG)
V1.3	02/01/2024	85%	Secure intra and extra CECCM components communications and Trust in AC^3	Adlen Ksentini (EUR), Ayoub Mokhtari (EUR), Abd Elghani MELIANI (EUR), Sofiane Messaoudi (EUR),
V1.4	31/01/2024	90%	Prepared the initial full draft	Sofiane MESSAOUDI (EUR)
V1.5	20/02/2024	95%	Received internal reviewers' feedback	Vrettos MOULOS (UPR) & Elias DRITSAS (ISI)
V1.6	28/02/2024	100%	Prepared the final draft	Adlen Ksentini, Ayoub Mokhtari, Sofiane MESSAOUDI, Abd Elghani MELIANI (EUR); John Beredimas, Athanasios Kordelas (CSG); Souvik Sengupta, Ali Nikoukar (IONOS)

Disclaimer

Copyright message

Table of Contents

List of Figures

List of Tables

Glossary of terms and abbreviations used

Abbreviation / Term	Description
2FA	Two Factor Authentication
AC³	Agile and Cognitive Cloud edge Continuum management
AAD	Azure Active Directory
AD	Application Developer
AES	Advanced Encryption Standard
AG	Application Gateway
AI	Artificial Intelligence
API	Application Programming Interface
ADC	Application Delivery Controller
CA	Certificate Authority
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CEC	Cloud Edge Continuum
CECC	Cloud Edge Computing Continuum
CECCM	Cloud Edge Computing Continuum Manager
CFN	Compute First Networking
CGI	Citrix Gateway Interface
CLI	Command-Line Interface
CRUD	Create, Read, Update, and Delete
DAC	Discretionary Access Control
DAPS	Dynamic Attribute Provisioning Service
DDoS	Distributed Denial-of-Service
DM	Data Management
DoS	Denial-of-Service
DTM	Dynamic Trust Monitoring
EDC	Eclipse Dataspace Components
ELK	Elasticsearch, Logstash, Kibana
FHS	Federation Hosting Service
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HaDEA	Health and Digital Executive Agency

HIPAA	Health Insurance Portability and Accountability Act
HSTS	HTTP Strict Transport Security
HTTP	HyperText Transfer Protocol
HTTPS	HTTP Secure
IdP	Identity Provider
IDSA	International Data Spaces Association
IDS-RAM	International Data Spaces Reference Architecture Model
InfP	Infrastructure Provider
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Tokens
KPI	Key Performance Indicator
LCM	Life-Cycle Management
LDAP	Lightweight Directory Access Protocol
LMS	Local Management System
MAD	Microsoft Active Directory
MFA	Multi-Factor Authentication
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MTLS	Mutual TLS
NB	North Bound
NBI	Northbound Interface
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
OIDC	OpenID Connect
OTP	One-Time Password
PaaS	Platform as a Service
ParIS	Participant Information Service
PAP	Policy Administration Point
PCI DSS	Payment Card Industry Data Security Standard
PDP	Policy Decision Point

PEP	Policy Enforcement Point
QoS	Quality of Service
RADIUS	Remote Authentication Dial-In User Service
RBAC	Role-Based Access Control
RTO	Recovery Time Objectives
S3	Simple Storage Service
SDWAN	Software-Defined Wide Area network
SIEM	Security Information and Event Management
SIIF	Standard for Intercloud Interoperability and Federation
SAML	Security Assertion Markup Language
SLA	Service Level Agreement
SLO	Service Level Objectives
SMS	Short Message Service
SNIP	Subnet IP
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
STS	Security Token Service
SOA	Service-Oriented Architecture
TLS	Transport Layer Security
ToC	Table of Contents
URL	Uniform Resource Locator
VIP	Virtual IP
VPN	Virtual Private Network
WAF	Web Application Firewall
WP	Work Package
YAML	Yet Another Markup Language

1 Executive Summary

This document, "D2.5: Security and Trust Management for CECC" is a crucial output of the AC^3 (Agile and Cognitive Cloud Edge Continuum Management) project, funded under Horizon Europe's Research and Innovation Action program. It delves into the intricate realm of security and trust management within the Cloud Edge Computing Continuum (CECC) and its interactions with the application developer and the infrastructure, aligning with task T2.4's objectives entitled "Security and Trust Management for CECC". It should be noted that task T2.4 is the only task that addresses security and trust aspects within AC^3.

The integration of cloud, edge, and far-edge computing platforms within the CECC landscape has transformed the digital ecosystem. This convergence empowers end users to harness the strengths of each platform, leading to enhanced performance, scalability, real-time responsiveness, and cost-efficiency across diverse applications. Key benefits include reduced latency, improved application responsiveness, optimized resource allocation, enhanced bandwidth efficiency, and reinforced data privacy and security. The integration of these platforms creates a dynamic ecosystem that optimizes performance, resource management, and data privacy, setting the stage for operational excellence in the digital era.

AC^3 project involves the interaction of different stakeholders to run micro-service-based applications on top of a CECC infrastructure, where we can mention application developers, CECC Manager (CECCM) provider, and infrastructure providers. Establishing a zero-trust security principle among the stakeholders while establishing trust when handling the requested Service Level Agreement (SLA) of the application is a must to have for any complex ecosystem like that of AC^3. The complexity of the AC^3 model is well captured in D2.1, where an initial high-level architecture has been proposed, which details all the components involved in deploying and running micro-service-based applications, covering the functional blocks as well as the communication interfaces among these blocks. However, the envisioned architecture did not detail how these interfaces are secured and how trust for SLA management can be established. This deliverable fills this gap by proposing initial solutions for security and trust in AC^3.

Indeed, in this deliverable, we present a high-level framework for zero-trust security and trust management within the AC^3 project. It is important to note that the term "trust" in this deliverable is used as a key approach for security architecture, but it is also an approach to managing SLA established between the application developer and CECCM and between CECCM and the different infrastructure providers. The aim of the first usage of trust is to secure all the communication interfaces, both internal and external, to the CECCM, leveraging the AC^3 architecture proposed in D2.1. The second usage of the word trust focuses on building a trust model to allow CECCM to select the infrastructure provider to run a micro-service. The trust model relies on Blockchain and Smart Contracts to derive the reputation of each infrastructure provider. Our objective is to safeguard data and ensure trust within CECCM and its interactions (i.e., Interfaces) with the application developer and the infrastructure. We have developed a robust data management strategy, emphasizing data security throughout its lifecycle, from retrieval to storage and monitoring. Additionally, we have fortified security measures within the CECCM, ensuring operational security while adhering to strict SLA benchmarks. Trust is a paramount concern, and stakeholders can rely on the integrity and reliability of the CECC ecosystem.

The AC^3 architecture, detailed in Section 3, provides a holistic overview, emphasizing its dynamic nature designed for continual evolution. Section 4 delves into the robust security measures in AC^3, including a detailed analysis of requirements, proposed approaches, and a comprehensive security architecture. API Security, covered in Subsection 4.3, underscores the encryption, validation, policies, authentication, and authorization strategies, ensuring the utmost security in interactions.

Trust is a cornerstone in AC^3, as highlighted in Section 5. The Trust Model is expounded upon, with a specific focus on SLA Management using Smart Contracts. The detailed discussion encompasses SLA and Smart Contracts definition, Blockchain integration, and the overarching trust architecture. The workflow, tools, and frameworks employed in this process are thoroughly introduced. Additionally, Section 5 addresses Trust in

Data Management, emphasizing security measures of data both in transit and at rest. These sections reflect our unwavering commitment to building and maintaining trust in the AC^3 ecosystem.

In conclusion, this deliverable serves as a foundational step toward AC^3's future endeavors, embodying our dedication to security, trust, and innovation in federated computing. The exhaustive content, drawn from various contributors and meticulously detailed in this report, encapsulates the collective effort invested in shaping the AC^3 landscape. This framework is not static; it is designed to evolve. Insights from WP3 and WP4 will shape its future. Particularly, WP3 will refine the SLA model that will be established between the application developer and the CECCM provider.

2 Introduction

In the dynamic landscape of modern digital interconnectedness, where real-time data processing, low latency interactions, and seamless user experiences have become paramount, the integration of Cloud, Edge, and Far Edge technologies has evolved into a necessity. This strategic fusion serves as the foundation for developing a federated computing continuum, one that seamlessly harnesses the power of centralized cloud resources while harnessing the immediacy of edge devices. Beyond achieving the fluidity of data flow and processing, this integrated approach accommodates the voluminous data inflow emanating from diverse sources.

AC^3 aims to define a novel CECC management framework that will play a critical role in supporting scalability and agility when managing CECC resources and emerging applications that rely on the micro-service concept. In D2.1, a high-level architecture has been proposed to manage the life cycle of micro-service-based applications from their definition up to their deployment over the Cloud Edge Continuum (CEC), considering cloud, edge, and far-edge resources. The envisioned architecture introduces CECCM a key element that interacts with the application developers through the application gateway, runs AI/ML-based LCM management functions, such as application profiling, application placement, run time management, and enforces decisions on top of a federated CECC infrastructure. The proposed AC^3 architecture handles not only the computing and networking resources but also all data that an application may require through the integration of data as Platform as a Service (PaaS). In addition to interacting with the application developers, CECCM interacts with the infrastructure providers for control and data plane of computing and networking resources. The different infrastructure providers belong to a federation, a marketplace-like system, allowing the CECCM to select the infrastructure provider that hosts one or all microservices composing the application.

The diverse actors involved in the AC^3 ecosystem call for efficient, secure mechanisms that ensure that all transactions and communications among the actors are secured and well-protected. Furthermore, there is a need to provide the CECCM with SLA management mechanisms to build trust knowledge regarding the infrastructure providers and hence ease the selection of one provider over the other when deploying micro-service components. The initial version of the high-level architecture of AC^3 focused on introducing the different functional blocks and the interfaces between each block, leaving the security and trust aspects aside. This is where this deliverable steps in by leveraging the initial high-level architecture with security and trust components. Regarding the security part, the objective is to enforce the concept of zero-security by ensuring authorization and authentication between all the components, including the internal components, before establishing communication among them and encrypting all communication between CECCM and external entities, i.e., application developers and the infrastructure platforms. To this aim, three new entities have been introduced and detailed in this deliverable: Security Gateway, Identity Provider and Authorization server, and Security Policies Administration.

Regarding SLA management to build a trust model, the deliverables design a novel architecture that involves a third-tier trusted management entity that collects feedback from the application developers, Key Performance Indicators (KPIs) from CECCM, and infrastructure providers to detect SLA violations and build infrastructure provider reputation. To achieve this objective, an approach based on Blockchain and Smart Contracts is introduced in this deliverable.

In summary, this deliverable introduces two main contributions to the AC^3 architecture: (1) zero trust security for the CECCM by adding the necessary components and well-defining their functional roles; (2) a trust model to track and control the SLA by defining a trust architecture and the necessary algorithms to build infrastructure provider reputations using Blockchain and Smart Contracts.

2.1 Mapping AC^3 Outputs

The purpose of this section is to map AC^3 Grant Agreement commitments, both within the formal Deliverable and Task description, against the project's respective outputs and work performed.

Table 1: Adherence to AC³ GA Deliverable & Tasks Descriptions

AC³ GA Component Title	AC³ GA Component Outline	Respective Document Chapter(s)	Justification
DELIVERABLE			
D2.5 Report on security and trust management for CECC			
TASKS			
Task T2.4 security and trust management for CECC	**Task T2.4:** This task devises robust protocols and mechanisms to establish trust with the candidates to the resource federation before granting access, guarantee communication, resource, and data security within the federation, and revoke access from federation members if security breaches are observed. Further, mechanisms to verify and guarantee signed SLA between the different stakeholders involved in the federated infrastructure. On the other hand, this task will implement protocols and enablers to secure the communication channel and protect the CECCM from external attacks as the latter is exposing interfaces to the public (application developers).	Section 4, & Section 5	Secure the intra and extra CECCM components communications and establish the trust in AC³ Framework.

2.2 Deliverable Overview and Report Structure

In this section, a brief description of the deliverable's structure has been provided as follows:

- Section 3 recalls and summarizes the AC³ architecture as introduced in deliverable D2.1.
- Section 4 initially provides the security approach, mechanisms, protocols, and architecture in AC³. Then, it focuses on securing the APIs by introducing encryption, validation, policies, etc. Finally, it introduces data security at rest, in usage, and in motion.

- Finally, Section 5 presents the trust model in AC^3 between the CECCM, the Application Developer and the Infrastructure Providers (i.e., CECC) by shedding light on SLA management. Additionally, it introduces the trust of the data management.

3 AC³ Architecture

The AC³ consortium has defined in D2.1 a high-level architecture that aims to handle the life-cycle of micro-service-based applications on top of a federated CEC infrastructure. The proposed architecture relies on a key element, namely the CECCM, which has the critical role in handling the micro-service life-cycle employing AI/ML-based techniques. On one hand, it sustains and supports the application's required SLA. On the other hand, it optimizes the CEC infrastructure considering energy consumption as a key parameter. In this section, we will recall the core aspects of the functional architecture, as shown in Figure 1, which encompasses the user, management and CECC infrastructure planes. Key components such as the Application Gateway, Application and Resource Management, Adaptation and Federation Layer, Ontology and Semantic aware Reasoner, and Data Management will be highlighted, offering a concise overview of the CECCM functionalities.

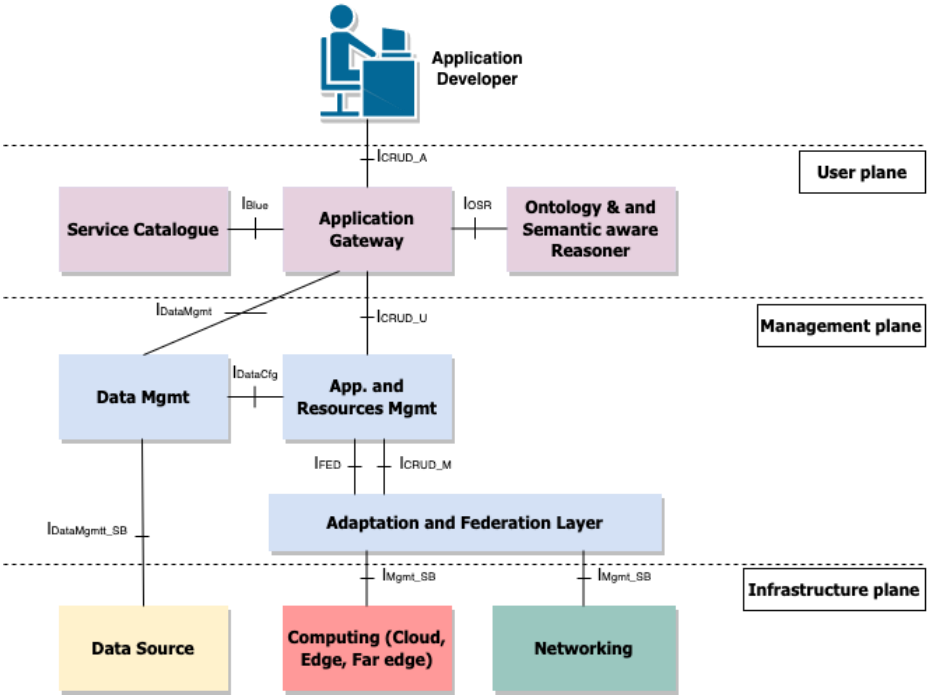

Figure 1. High-Level Architecture of AC³

3.1 Overall Architecture

The User Plane: This plane serves as the interface for application developers, encapsulating CECCM functionalities through the Application Gateway. It offers a simplified environment for developers to develop, deploy microservice-based applications and manage their life-cycle with intuitive interfaces for CRUD operations. Notably, in AC³, application descriptions and SLA can be expressed using intents. These intents are translated by the Application Gateway into machine-readable formats such as YAML or JSON, then forwarded to the management plane.

The Management Plane: Central to CECCM, the Management Plane orchestrates applications and manages CECC resources through AI/ML algorithms. It includes Application and Resource Management modules that handle application LCM, DM, and the CECC abstraction and federation, enabling seamless integration of diverse infrastructure resources. This plane consists of three key components : (i) Application and Resource Management that serves as a core module for managing both the LCM of microservice-based applications by handling the runtime of applications to guarantee SLA, and the diverse set of the resources within the CECC by interfacing with the federation and abstraction layer to select resources and monitor applications

behaviour; (ii) Data Management that manages accessing both to hot and cold data following the Gaia-X[1] procedures for data lakes and IoT data sources federated with AC^3 access; (iii) Abstraction and Federation Layer that acts as an interface bridging the management functions of the CECCM with the federated infrastructure. It abstracts the heterogeneous infrastructures and enables essential functionalities such as resource discovery and CRUD operations over the federated CECC infrastructure.

The Federated CECC Infrastructure Plane: Envisioned as a federated CECC infrastructure, it aggregates resources composed of public/private cloud resources, edge resources, far-edge resources, and networking resources. As detailed in D2.1, the federation aspect of AC^3 relies on a hybrid approach that combines elements from both the Gaia-X model for facilitating interoperability of services and securing cross-border data sharing and federation, as well as the National Institute of Standards and Technology (NIST)[2] model for efficient cloud resource allocation and management. Particularly, when deploying microservices-based applications across multiple infrastructure providers, Gaia-X facilitates the secure exchange of data across these providers. The establishment of trust among different AC^3 stakeholders is achieved through the utilization of linked data representation and verifiable credentials through the Gaia-X compliance web portal and trust anchors.

Adapting the IEEE SIIF architecture, the interaction between the CECCM and the federated resource infrastructures is enabled by the Federation Hosting Service (FHS) acting as an external entity to the CECCM. As the CECCM will be a member of the federation having permission to discover services offered by the resource infrastructure, the Resource Discovery module at the Adaptation and Federation layer allows the discovery of available services in the federation. FHS also allows resource infrastructures to join the federation and to register services such as NBI service and resource exposure service of the Local Management System. These registered services will be utilized by the adaptation and federation of the CECCM for resource management.

3.2 CECCM Key Components

Application Gateway: This component offers a gateway for developers, providing both GUI and API interfaces for CRUD operations. It facilitates the creation, deployment, monitoring, and removal of micro-service-based applications, translating human-expressed intents into machine formats.

Application and Resource Management: At the core of CECCM, this component employs AI-driven LCM for application deployment, runtime management, and resource optimization. It consists of five modules:

Firstly, the **AI-based LCM** orchestrates application life cycles, managing deployment, application resources, and runtime adjustments using AI algorithms. **The Monitoring module** collects diverse data from the CECC infrastructure, aiding in application and resource optimization. Meanwhile, the **AI-Application Profile** continuously updates application profiles by analyzing monitoring data, offering insights into application behavior. Similarly, the **AI-based CECC Resource Profile** predicts resource utilization and performance, aiding in optimized resource management. Lastly, the **Decision Enforcement module** executes decisions derived by the AI-based LCM, deploying and managing microservices across infrastructure nodes.

Adaptation and Federation Layer: Serving as an intermediary, this layer stands as an interface between the CECCM management functions and the federated infrastructure, facilitating resource discovery, selection, and interfacing with Local Management Systems (LMS). It streamlines CRUD operations across heterogeneous infrastructures.

Ontology and Semantic aware Reasoner: As an intelligent reasoning engine, this component utilizes semantic web technologies to interpret and adapt policies for managing microservice-based applications. It transforms human-defined policies into dynamic, context-aware rules, optimizing the application lifecycle.

[1] https://gaia-x.eu/

[2] https://www.nist.gov/

Service Catalogue: Centralizes application blueprints and metadata, providing developers with a comprehensive repository to explore, access, and manage available services. It supports CRUD operations for application blueprints and assists in tracking ownership and essential metadata.

4 Security in AC³

As commonly admitted, there are no universal guidelines for securing software products. The security framework of a system is primarily influenced by its specific software architecture. As stated earlier, in AC³, the suggested architecture consists of three distinct planes: the user plane, the management plane, and the CECC plane. Each of these planes incorporates its own components and relies on clearly defined interfaces for communication with the other ones. Well understanding this information yields us towards recognizing the distinct challenges involved in securing software with multiple services compared to securing a monolithic system. The diverse nature of components in a multi-service software architecture requires a nuanced approach to security because, first, the more extensive the attack surface, the greater the susceptibility to potential security threats. In a monolithic application, internal components communicate within a single process. In contrast, within a multi-services architecture, these internal components are typically crafted as distinct, standalone units, transforming in-process calls among them into remote calls, and each component independently accepts requests or has its own entry points. With an expanding number of entry points, the attack surface widens. Managing the security design for the components becomes a fundamental challenge, requiring equal protection for each entry point to every component. The overall security of the system is only as robust as the security measures implemented at its most vulnerable entry point.

Furthermore, distributed security screening has the downside of potentially degrading the overall system performance. Indeed, in a multi-service baked application deployment, unlike a monolithic application where security screening is performed once, each component conducts independent security screenings. The multiple security screenings at each component's entry point may appear redundant. Additionally, during request validation, connections to a remote Security Token Service (STS) may be necessary. These repetitive distributed security checks and remote connections can introduce significant latency, leading to a noticeable degradation in system performance. Some address this challenge by opting to trust the network, bypassing security checks at each service. However, over time, relying solely on network trust has become recognized as an antipattern, prompting a shift in the industry towards adopting zero-trust networking principles. With zero-trust networking, security measures are implemented much closer to each resource in the network this concept introduced by John Kindervag [1] in 2010, considers simply that there is no trusted network traffic, this implies the necessity to access all resources securely, irrespective of their location, while minimizing privileges to the essential level and this also means that any multi-services software security design should carefully consider overall performance implications and take proactive steps to mitigate potential drawbacks. Google's BeyondCorp [2] is often considered one of the pioneering implementations of the zero-trust architecture for enterprise security. Launched in 2014, and by shifting access controls from the network perimeter to individual users, BeyondCorp enables secure work from virtually any location without the need for a traditional Virtual Private Network (VPN). In 2020, the NIST issued the guideline NIST SP 800-207 [3], highlighting that no resource within a system should be inherently trusted. Secure communication is mandated, irrespective of the network's location. The application should be subject to interruption and inspection, with a thorough examination of all user-associated contexts available on demand. Dynamic and stringent authentication and authorization processes must be enforced before granting access to any resources. Another challenge of securing multi-service software is that the distributed nature of services makes it harder to share user context. In a monolithic application, all internal components share a common web session, enabling easy retrieval of anything related to the requesting party (or user). However, in microservices or a Service-Oriented Architecture (SOA) based architecture, this convenience is not present. Microservices do not share much information (or only a limited set of resources), and the user context must be explicitly passed from one microservice to another. The challenge lies in establishing trust between two microservices so that the receiving microservice accepts the user context passed from the calling microservice. In this situation, it is crucial to have a mechanism to verify that the user context passed among microservices hasn't been intentionally altered.

A widely adopted approach to facilitate the exchange of user context among microservices is the utilization of JSON Web Tokens (JWT)[3]. This method employs a JWT as a JSON-based message, securely conveying a predefined set of user attributes from one microservice to another in a cryptographically secure manner.

4.1 Requirements and Proposed Approach

While the methodologies may undergo alterations, the overarching objective remains constant. Embracing foundational principles is imperative in the development of any robust security design. There is no perfect or unbreakable security. The level of concern one should have about security is determined not just by technical factors but also by economic considerations. Regardless of the situation, it is crucial to stick to basic security principles. Even if you cannot anticipate specific security threats, adhering to these fundamentals is essential for safeguarding a system against potential risks. In this section, we will recall the principles and the essential security fundamentals that a secured system must target.

Authentication: Authentication involves identifying the requesting party, serving as a safeguard against spoofing to ensure system protection. The requesting entity may be a system, such as a service, or a system acting on behalf of a human user or another system. Prior to devising a security design for a particular software, it is essential to ascertain the audience because the choice of the authentication method is based on it.

Integrity: When data is transmitted from a client application to a service or between services, the security of the communication channel becomes crucial. Depending on the strength of the chosen channel, there is a risk of interception by an intruder who may manipulate the data to their advantage. For instance, considering a microservices system in which data is shared among various modules if a malicious actor successfully compromises the communication channel, there is a risk of manipulating the data, causing disruptions in the system. Systems that prioritize integrity recognize this potential threat and implement measures to allow the recipient to detect and discard altered messages.

The primary method for ensuring the integrity of a message is through signing. When data is transmitted over a secure communication channel, such as one implemented with Transport Layer Security (TLS), integrity is inherently protected. This cryptographic technique involves digitally signing the transmitted data to verify its authenticity and ensure it has not been tampered with during transit. While HTTPS (HTTP over TLS) is a commonly employed protocol for securing communication between microservices, it is important to note that various communication protocols can be used in a microservices architecture. For instance, besides HTTPS, microservices may utilize other protocols like Message Queuing Telemetry Transport (MQTT) for efficient communication in IoT applications or Simple Object Access Protocol (SOAP) for specific web service interactions. Each of these protocols may have its own security mechanisms to guarantee the integrity of transmitted data. Therefore, the choice of communication protocol depends on the specific requirements of the microservices application, and the implementation of secure communication channels is a crucial consideration regardless of the protocol selected.

Non-repudiation: Non-repudiation is the assurance that someone cannot deny the validity of something. Non-repudiation is a legal concept that is widely used in information security and refers to a service that provides proof of the origin and integrity of data. Non-repudiation makes it very difficult to successfully deny who/where a message comes from, as well as the authenticity and integrity of that message. Digital signatures (combined with other measures) can offer non-repudiation. It's crucial to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or when sending the communication in the first place.

Confidentiality: Data integrity is the assurance that digital information is uncorrupted, complete, and consistent. However, sometimes, depending on the security of the chosen communication channel, there is a risk of interception by intruders who could gain unauthorized access to the data and gain access and view the confidential information. Data confidentiality involves protecting sensitive information from

[3] https://www.json.org/json-fr.html

unauthorized access and ensuring that only individuals with proper authorization can access and view confidential data. Encryption, access controls, and secure storage ensure that third parties cannot easily decrypt and view information they are not allowed to.

Availability: The whole point of building any kind of system is to make it available to its users. The overall architecture, not just security, influences system stability, but security has a vital role in making a system constantly available to its legitimate stakeholders. In a multiservice deployment, with many entry points (which may be exposed to the internet), an attacker can execute a Denial-of-Service (DoS) or a Distributed Denial-of-Service (DDoS) attack and take the system down. Defenses against such attacks can be built on different levels. On the application level, a reasonable approach is to reject a message (or request) as soon as it is considered illegitimate.

Authorization: While authentication helps a system to learn about the user or the requesting party, authorization on the other hand determines the actions that an authenticated user can perform on the system. In a typical microservices deployment, for example, in microservices-based applications, authorization is typically implemented at two levels: firstly, at the entry point to the application deployment, which could potentially be intercepted by a gateway; and secondly, at each individual microservice level. This ensures that authorization is enforced both at the overall application entry point and between the application microservices.

4.2 Security Architecture

In this section, we propose several improvements to the AC^3 architecture, as shown in Figure 1, following the key criteria for creating an efficient and secure system with the aim of achieving a zero-trust security model. The CECCM has external and internal communication that both need to be secured. The external communications involve interactions between the CECCM and external entities such as application developers and the NBIs of the different infrastructures LMS, while the internal communications consist of the interactions between the CECCM components. While the security of the communication and agreements among the federation actors is out of the scope of AC^3, securing the communication between CECCM and the infrastructure providers participating in the federation should be addressed. As stated earlier, AC^3 uses the NIST model for the federation, hence relying on the trust and security mechanisms already specified by NIST. From our perspective, securing both north/south communications between users and the CECCM or between CECC federated infrastructure and the CECCM and east/west communications among services within the CECCM can significantly reduce the vulnerabilities within the CECCM architecture. A leveraged version of the AC^3 architecture featuring security is illustrated in Figure 2, which includes several novel components aiming to achieve zero trust security architecture. These components are the Secure Gateway, Indentity provider and authentication server, and Security Policies Administration. Their functions and roles will be detailed in the next section.

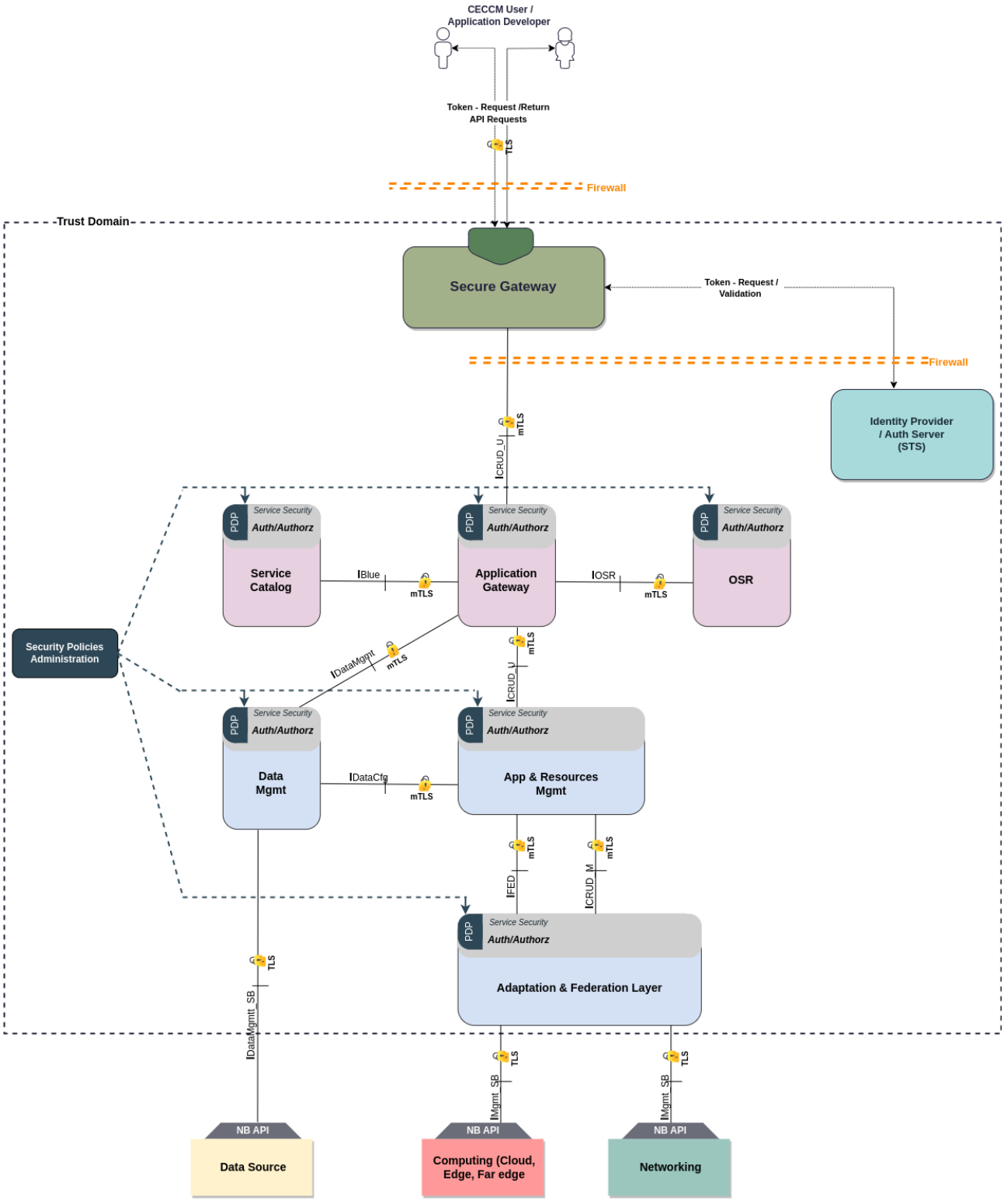

Figure 2. Security Architecture Overview

4.2.1 Secure User-CECCM Communications

The Secure Gateway depicted in Figure 2 acts as a policy enforcement point (PEP), authenticating, authorizing, and intercepting all end-user requests going to the AG API to enforce access-control policies, and then it dispatches the requests with the user's context to the AG API implementation. The Secure Gateway's

role is to ensure that only CECCM users with trusted certificates can access the APIs and that only those requests are directed to the upstream services.

For end-user authentication, commonly utilized approaches include certificate-based authentication[4], which safeguards an API at the CECCM entry point. Access to other services is conditional on the customer's possession of a valid certificate issued by a trusted entity such as an identity provider. The alternative approach involves employing OAuth-2.0–based[5] access delegation. OAuth 2.0, an authorization framework facilitating delegated access control, is the preferred method for safeguarding APIs when one system intends to access an API on behalf of another system or user. In this context, the secure gateway's responsibility is to authenticate the OAuth 2.0 security tokens accompanying each API request. These tokens serve as a representation of both the third-party application and the user who granted access to the third-party application to interact with an API on their behalf.

In addition to identifying the requester during the authentication process, the Secure Gateway has the ability to enforce authorization by applying access control policies. More detailed access control policies are then implemented at the service level by the respective service itself. Irrespective of the chosen authentication and authorization approach, the primary function of the Identity Provider/Auth Server depicted in Figure 2 remains unchanged—producing tokens or certificates to the Secure Gateway for user's authentication and authorization.

After verifying the integrity of the connection, the secure gateway forwards the requests with user context to the AG API implementation to be then redirected to the appropriate CECCM service, e.g., Application and Resource Management service. The user context includes fundamental details about the end user, and the client context provides information about the client application. This information can be utilized by the CECCM services for service-level access control. For instance, consider a scenario in which application developer <A> wants to consume data of another deployed application <App1>, the application developer <A> context will be used to limit his service access to only consuming the <App1> data and to prevent access to <App1> LCM. To transmit the context to CECCM services, there are a couple of options: either pass the user context in an HTTP header or create a JWT containing the user data. While the first option is straightforward, it raises trust concerns when one service transmits the user context in an HTTP header to another service, as the second service lacks assurance that the user context remains unaltered. Opting for the second approach with JWT provides confidence that a man-in-the-middle cannot modify its content without detection, as the issuer of the JWT signs it. Therefore, the second approach is considered more secure. It's worth noting that the communication between the Secure Gateway and the AG API (if not implemented as a single component) necessitates mTLS authentication to ensure channel security.

4.2.2 Secure CECCM Service-Service Communications

Similar to user-CECCM communications, ensuring authentication and authorization between services is essential to minimize vulnerabilities and security threats. In general, security models designed to safeguard service-to-service communication need to account for communication channels traversing trust boundaries and the nature of the communication itself, whether it's synchronous or asynchronous. In our specific case, details about the communication method are not yet available. Regarding trust domains, we are considering one for the CECCM services and another for the federated infrastructure for now.

Authentication: For authenticating inter-CECCM components communications, three commonly used approaches are generally employed: trusting the network, mutual TLS (mTLS), or JWTs (JSON Web Tokens).

Trusting the network approach entails no explicit security enforcement in service-to-service communication. Instead, the model relies on network-level security, where the assurance lies in preventing attackers from intercepting communications between services. Additionally, adopting a trust-the-network approach would require deploying the CECCM within the same infrastructure and network, potentially leading to constraints

[4] https://www.ibm.com/docs/en/ztpf/2020?topic=certificates-certificate-based-authentication
[5] https://oauth.net/2/

on implementation. Hence, this approach may not be the most suitable for our case. In contrast, the zero-trust network approach offers an alternative viewpoint to trust-the-network one. This model assumes a consistently hostile and untrusted network environment, rejecting any assumptions trusting the network. Every request within this approach must pass through authentication and authorization at each CECCM service (component) before being accepted for further processing.

One method for authenticating CECCM inter-service communication is mTLS, which stands as a widely adopted approach for securing service-to-service interactions. Each service within the deployment is equipped with a public/private key pair, utilizing this pair for authentication when communicating with recipient services through mTLS, ensuring mutual identification between communicating services.

The third approach for securing service-to-service communications is JWT. In contrast to mTLS, JWT operates at the application layer rather than the transport layer. A JWT serves as a container capable of transporting a set of claims from one location to another. These claims can encompass a variety of information, such as end-user attributes, end-user entitlements (defining what the user can do), or any data that the calling CECCM service wishes to convey to the recipient service. The JWT encapsulates these claims and is signed by the issuer of the JWT, which can be the STS or the calling CECCM service itself.

In our scenario, to capitalize on the strengths of the two previous mechanisms, we have opted for a hybrid approach. This involves combining mTLS for encryption and authentication with JWT for transmitting essential information between services, such as user details or authorization levels.

With mTLS, communication is encrypted, ensuring the confidentiality and integrity of the transmitted data. Mutual authentication is achieved as both the calling service and recipient service authenticate each other through their presented certificates.

To convey authorization-related information between the communicating services, we utilize JWT. When a calling service authenticates itself using mTLS, it can include a JWT in the request headers or as part of the payload. This JWT encapsulates various information, such as roles, permissions, or other pertinent claims. Upon receiving the JWT, the recipient service can verify its signature using the public key associated with the issuer, ensuring the integrity of the token and the authenticity of the claims.

Authorization: When considering authorization in a typical multi-service CECCM, service-level authorization is necessary to provide each service with a higher degree of control over enforcing access-control policies according to its specific requirements. The role of the Policy Decision Point (PDP) integrated within each CECCM component (see **Error! Reference source not found.**) acts as a store for policies. These policies are c entrally defined at the Policy Administration Point (PAP) and locally evaluated (at each CECCM component level). To receive policy updates from the centralized PAP, each CECCM service acts as an event consumer and subscribes to the relevant PAP event. Upon receiving an event, the service retrieves the corresponding policy from the PAP and updates its embedded PDP accordingly to be used in CECCM service-level authentication and authorization (e.g., updating access-control policy for DataMgmt Service).

The precise definition of the centralized PAP definition depends on the CECCM implementation. In the illustrative example depicted in Figure 3, where the CECCM is implemented using a microservices approach, we can leverage the service mesh pattern for enforcing security at each microservice while decoupling the security from the microservice logic. In this context, the control plane provides the centrally defined security policies acting as PAP.

4.2.3 Secure CECCM-Federated Infrastructure Communication

In a typical deployment of multiservice software, it is common to encounter scenarios involving multiple trust domains. By "trust domain," we refer to a collection of services that place trust in a single identity provider. From a security standpoint, when one service communicates with another within the same trust domain, both services may trust the same identity provider or certificate authority. Leveraging this mutual trust, the recipient service can verify a security token received from a calling service.

In our specific case, there are instances where the CECCM needs to establish connections with the NB APIs of LMS within the federated infrastructure. In such a case, the different infrastructures LMSs need to provide certificates delivered by a trusted public certificate authority to ensure the authenticity of each infrastructure. In addition, the policy decision point or the security layer of the CECCM adaptation and federation component can encrypt a secret key using the LMS public key and send it to the infrastructure LMS for further communication tunnel encryption. This process is illustrated in Figure 3.

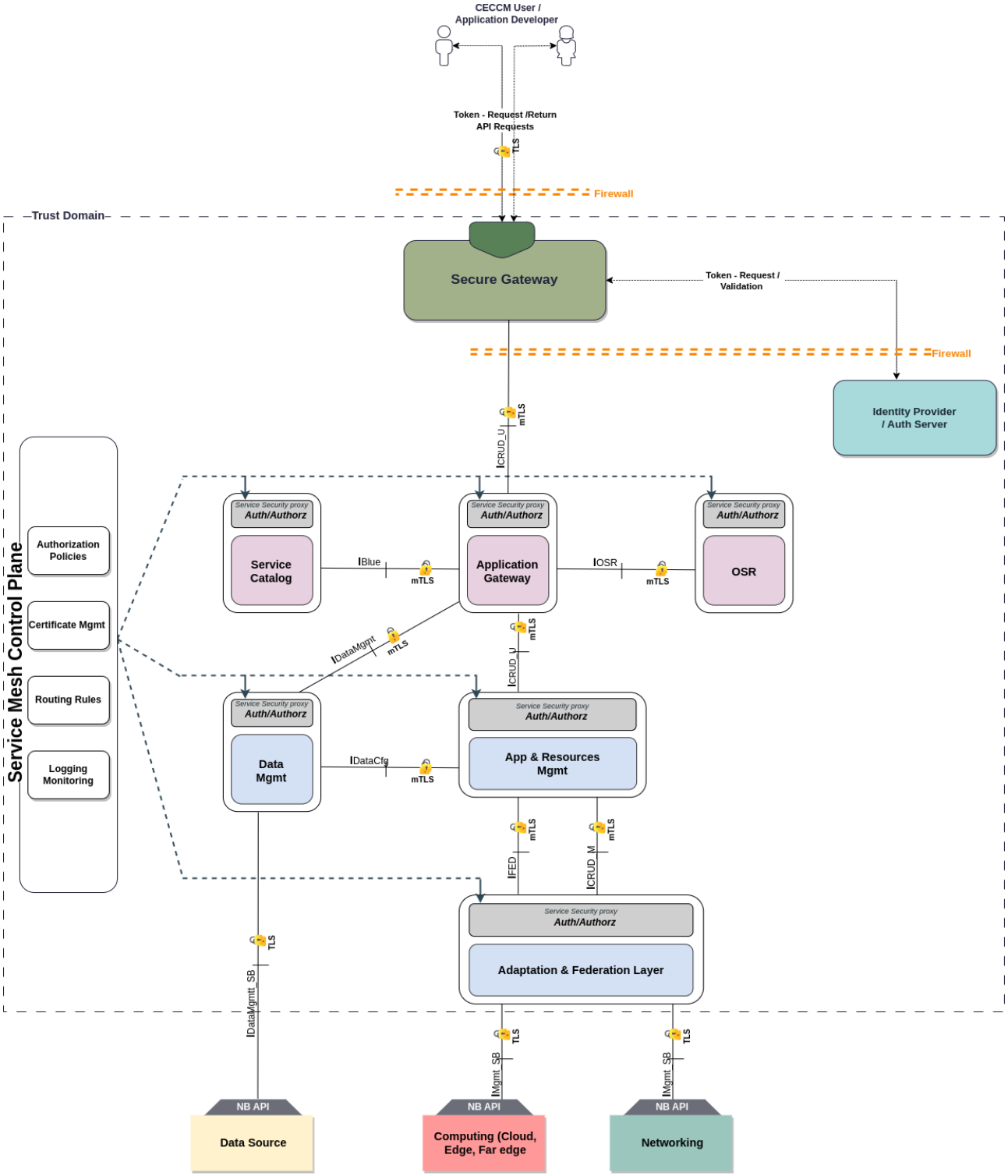

Figure 3. Security Architecture Instantiation in the Case of CECCM Microservices based Implementation

4.2.4 Workflows

To improve the comprehension of the envisioned methodology conducted in AC^3, this section includes illustrative diagrams delineating the secure communication workflow in three specific scenarios: communication between the application developer and the CECCM and within CECCM internal services.

- **Communication with the Application Developer:**

Consider a scenario where a developer seeks to deploy an application within the infrastructure managed by the CECCM. As outlined in Deliverable 2.2, the deployment process entails the developer furnishing the application's definition to the CECCM. In addition to this, for initial deployment requests, the developer is required to submit its credentials to the CECCM through the secure gateway.

Subsequently, upon receiving these credentials, the secure gateway interacts with the identity provider to either generate or retrieve the user certificate. This certificate serves as an authentication mechanism for subsequent interactions. Once obtained, the secure gateway initiates a request to the identity provider to acquire a JWT token containing the user context, encompassing roles and privileges.

The secure gateway then scrutinizes the user's authorization level in comparison to the request at hand. If there is a match, the secure gateway proceeds to forward the request, along with the user context, to the user plane for enforcement. In cases where the user's authorization level does not align with the request, the secure gateway rejects the request. This intricate process ensures the secure and authenticated deployment of applications within the CECCM infrastructure. These sequences are illustrated in Figure 4.

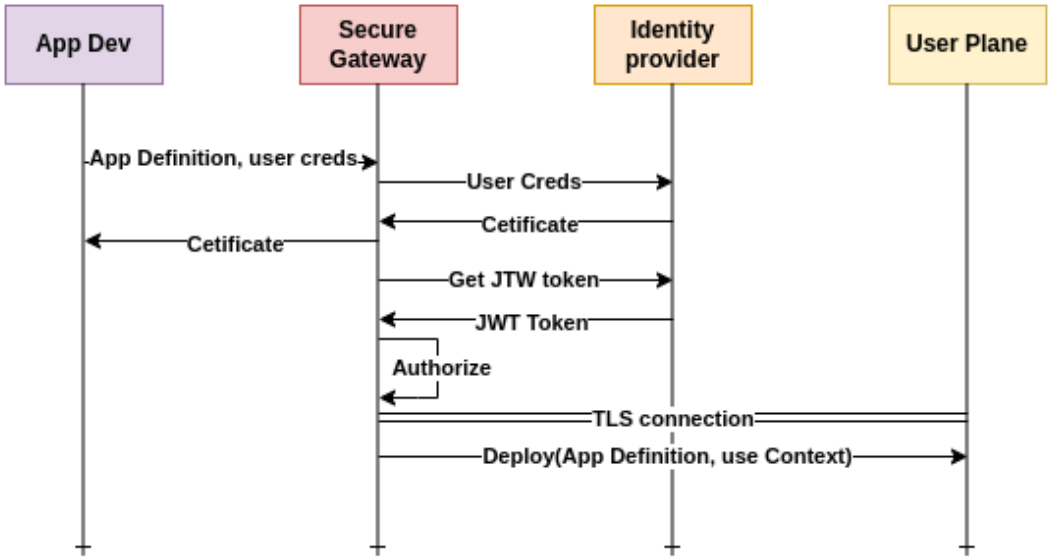

Figure 4. Secure Communication Workflow between the Application Developer and the CECCM

- **Internal Services Communication:**

In the context of internal communication between services, where mutual authentication is imperative, let us consider an illustrative scenario involving the application gateway and the service catalog component. In this instance, the application gateway seeks access to blueprints and data sources from the service catalog.

To initiate the process, the application gateway submits a request to the service catalog, presenting its certificate issued by the identity provider to substantiate the request for blueprints. Within its authentication layer, the service catalog meticulously validates the sender's identity, confirms the request's alignment with the specified authorization level, and, upon successful verification, fulfills the request. The service catalog responds by providing both the requested blueprints and its own certificate to the application gateway.

Upon receiving the blueprints and the service catalog certificate, the application gateway undergoes an authentication confirmation process, leading to the establishment of a mTLS session between the two components. With this authentication in place, the application gateway gains the privilege of making subsequent requests for data sources and blueprints without undergoing repeated authentication. It is crucial to note that the authorization process is reiterated for each request, ensuring the ongoing security of the communication.

The conceptualization of this process is visually articulated in Figure 5, offering a graphical representation of the interaction dynamics between the application gateway and the service catalog component.

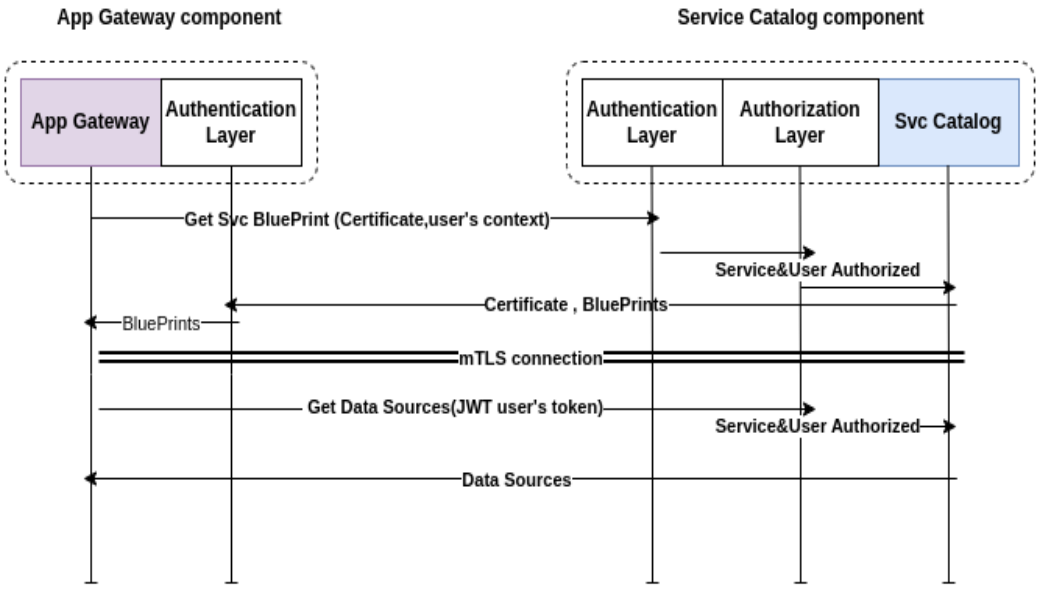

Figure 5. Secure Communication Workflow between CECCM Services

- **CECCM - Infrastructure LMS Communication**

As mentioned previously, the security of communications and agreements among the federation actors is out of the scope of AC^3. However, communications between the CECCM and the various LMSs need to be secured to at least ensure authenticity of the LMS and confidentiality of the information exchanged. The following workflow diagram describes an example where the Adaptation Agent initiates a secure connection with an LMS for an application onboarding scenario. In order to authenticate the LMS and trust its certificate, we assume that CECCM trusts the public certificate authority that signed the LMS certificate. This certificate not only authenticates the LMS, but also uses its public key for session key encryption. Once a secure session is established, the adaptation agent can interact with the LMS's NBI endpoint.

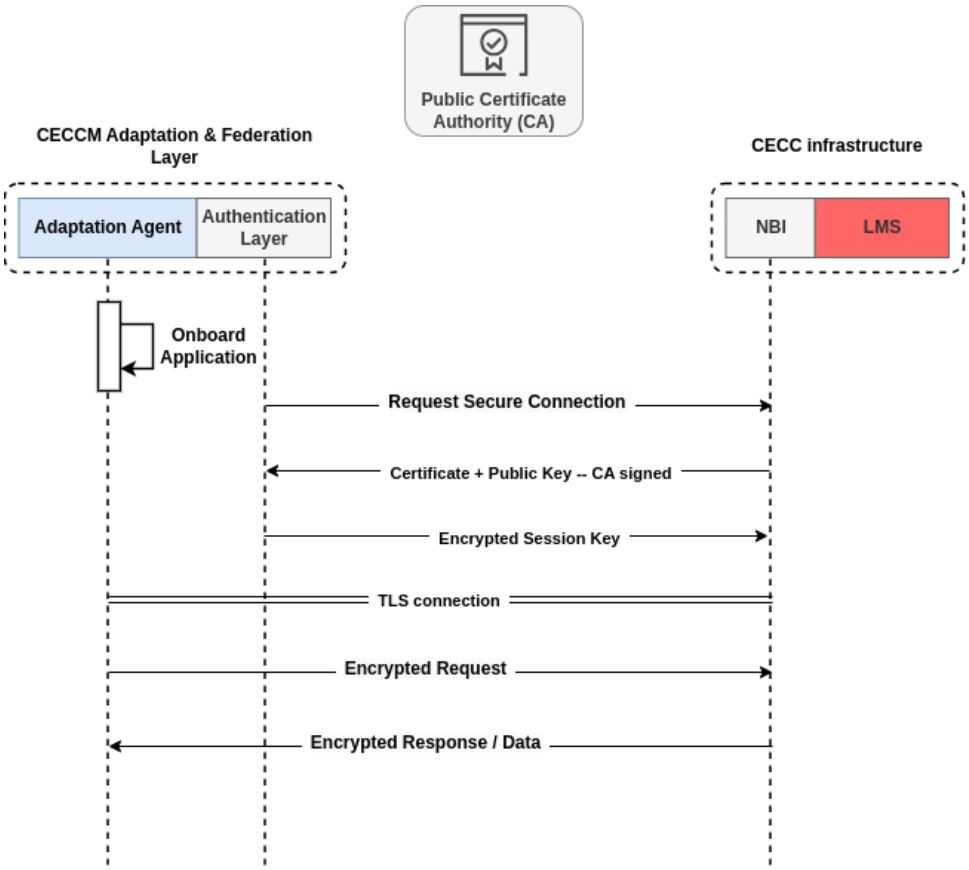

Figure 6. CECCM – Infrastructure LMS Secure Communication

4.3 API Security

APIs, or Application Programming Interfaces, are sets of rules, protocols, and tools that allow different software applications or systems to communicate with each other. APIs play an important role in protecting sensitive data by enforcing access controls, authentication, and encryption, ensuring that only authorized entities can access and transmit confidential information securely.

APIs are critical when it comes to security and trust in any well-designed system, including but not limited to AC^3. First and foremost, the most typical pattern for microservices, similar to the ones that will run in the AC^3 CEC, are to directly communicate with each other through well-defined APIs. Secondly, all user and device interactions, such as listing available services, managing apps, resources and data, performing lifecycle actions, registering new data or compute resources to the federation, and more, are expected to be performed through well-defined APIs rather than ad-hoc editing of configuration files or manually executed CLI commands. Towards this end, it is critical to ensure that said APIs are well-protected.

The remainder of this section examines in further detail the NetScaler API Gateway capabilities and how it can help secure any API deployed in the AC^3 architecture, including but not limited to the Secure Gateway, the Application Gateway and NB APIs of Local Management Systems.

4.3.1 Encryption and TLS Offload

TLS offloading, commonly referred to as SSL offloading, typically refers to the insertion of an intermediary proxy in front of a service or application. The intermediary proxy exposes a secure protocol to external services/application clients, to allow for encrypted communication for data in transit on untrusted networks, such as the Internet. Internally, the intermediary proxy decrypts incoming client requests, prior to sending them to the application, as well as encrypts app responses prior to propagating them to the client.

A typical TLS offload deployment, coupled with Load Balancing capabilities is illustrated in Figure 7. NetScaler intercepts all encrypted application traffic to an external Virtual IP (VIP), load balancing requests to application servers using an internal Subnet IP (SNIP).

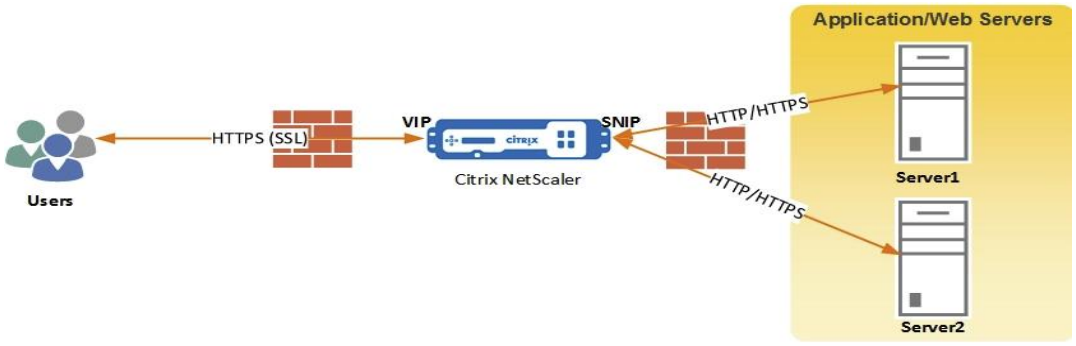

Figure 7. Typical NetScaler TLS Offload Deployment

NetScaler CPX can provide for TLS offload of both North-South traffic, which commonly refers to incoming traffic from the Internet, as well as East-West traffic, which commonly refers to communication between microservices. External CECCM interactions, such as APIs exposed to application developers or CECCM infrastructure owners are typical examples of North-South traffic, for instance the Application Developer to Secure Gateway traffic as illustrated in Figure 3. Intra-CECCM components communication, such as Service Catalog to Application Gateway and Application Gateway to OSR, would be a typical example of South-West traffic.

NetScaler provides industry-leading TLS offload performance and security. This is beneficial to both developers and operators of the AC^3 CEC. From a developer perspective, one can focus on API semantics and design, rather than the security implications of selecting a proper TLS version, the performance implications of choosing an antiquated cipher, appropriate encryption key length sizes, validation of certificate bundles, and other complexities related to TLS configuration. From an operator and service manager perspective, NetScaler can enable uniform management of TLS settings across the entire CECC, including but not limited to actions such as distribution of certificate bundles, certificate revocation, addition or removal of TLS ciphers, withdrawal of support for vulnerable TLS versions (i.e. TLS/1.1) [4] and more. For instance, when an application developer or operator deploys a NetScaler API proxy, they can easily choose a “High Security Profile”, which is commonly defined and maintained once for all services and ensures the highest possible A+ security rating from SSL Labs2.

4.3.2 API Validation

One key element of API security is the validation of incoming requests. Similarly, to any web applications, APIs can be compromised by malicious attackers or client implementation bugs that inadvertently expose issues with the API implementation. These issues include, but are not limited to, lack of value validation (i.e. accepting an integer value for a parameter that is of enum type), accepting HTTP methods that are not available for a specific endpoint (i.e. PUT/POST methods for an endpoint that only accepts GET), accepting API requests that lack mandatory parameters, accepting requests with an optional parameter that is not specified in the API specification and more. Lack of appropriate API validation will occasionally lead to the API implementation behaving in either an undefined or outright erroneous manner. For instance, lack of appropriate input validation may simply lead to a “Bad request” or “Processing failure” type of error, which is mostly harmless. However, more serious errors can also occur. Lack of method validation, i.e. accepting POST/PUT methods for a GET endpoint, may lead to a by-design immutable object being overwritten by a malicious or simply misbehaving client. In a worst-case scenario, lack of validation can expose the API implementation to either Denial of Service attacks, leading to a crash due to unexpected input, or serious security flaws.

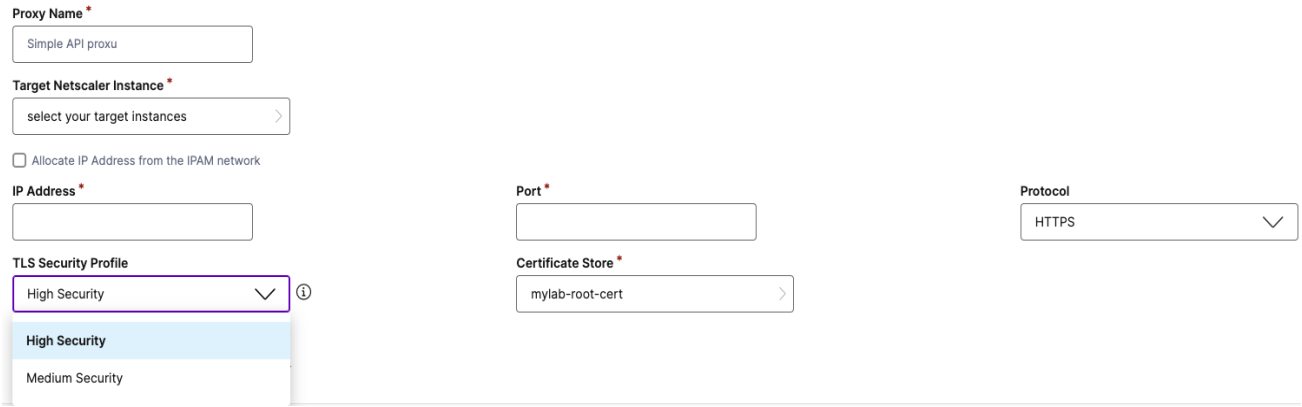

Figure 8. NetScaler API Proxy Definition for A+ SSL Security

API validation is one of the most basic capabilities of NetScaler when operating as an API security/gateway. Typically, the API developer or operator will import an OpenAPI schema specification to the NetScaler so as to automatically get the following:

Endpoint Whitelisting: Only API endpoints defined in the schema will be accepted; other endpoints will be dropped.

Endpoint Method Validation: For each endpoint only, the HTTP methods defined in the schema are supported. For example, a typical API convention calls for POST /resources to create a new resource but GET|PUT /resource/{id} to retrieve or modify details of an already existing resource. NetScaler API validation ensures that any POST /resource or GET|PUT /resources/{id} will be discarded, without potentially compromising the underlying implementation.

HTTP Header Validation: Typical HTTP header checks, including but not limited to header length and mandatory headers, are implemented.

Query String Parameter Validation: Any parameter defined in the request query string arguments is checked against the list of mandatory and optional parameters defined in the schema file. Value validation is also being performed to verify that values for integer parameters fall within the range, values for enum parameters are within the list of acceptable ones, and string parameters are of the appropriate length.

URL Path and HTTP Body Validation: Depending on the schema, parameters for API requests may be defined in the query string, URL path or HTTP body. The NetScaler API validation supports the same checks outlined above, regardless of the parameter location.

4.3.3 API Policies

NetScaler allows several different policies to be attached to an API profile. While these policies are not specific to API request handling, they can enable for additional protections and capabilities related to API security.

Authentication: HTTP Basic Authentication or OAuth2 access tokens are supported for API authentication. Note that authentication is optional and maybe implemented by a different layer.

Authorization: in case authentication is enabled, an authorization policy using a set of claims MUST be defined, even if this includes an empty set of claims (implying that all authenticated users have access to the respective resources).

WAF: Web Application Firewall, also known as WAF for brevity, provides protection against known and unknown common types of attacks, including but not limited to buffer overflow, cookie security attacks, web form security attacks, SQL injection, cross-site scripting, malicious or malformed payloads and more.

Bot Protection: NetScaler employs a number of techniques to identify if traffic originates from actual end user and applications or potentially malicious bots and can treat said traffic differently, i.e. apply rate limit, silently drop or deny requests, introduce CAPTCHAs etc. Typical techniques include device fingerprinting, IP reputation, rate limit violations and bot traps.

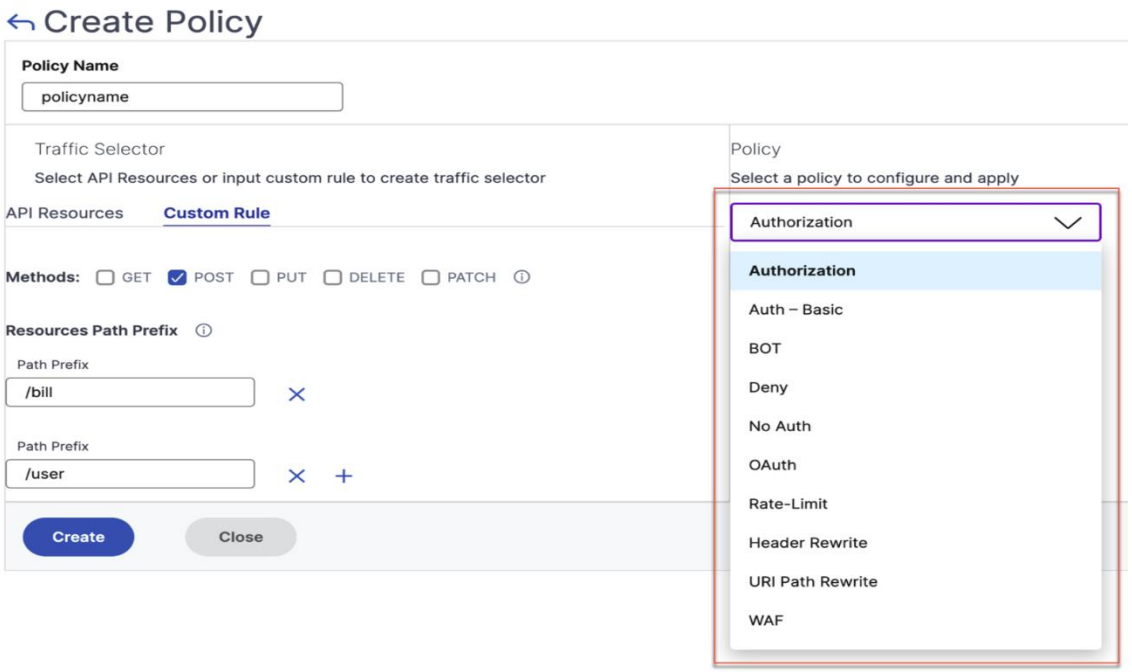

Figure 9. NetScaler API Policy Types

Rate Limiting: Rate limiting can protect APIs against surges of traffic, due to misbehaving or malicious clients, by either dropping / resetting excess requests, redirecting to another URL or most typically responding with a "429 too many requests" HTTP error message.

4.3.4 Authentication and Authorization

Authentication and authorization, commonly referred to as AuthN and AuthZ respectively, are two vital information security processes used to protect systems and information in any large infrastructure with multiple stakeholders such as AC^3. While complimentary, they are distinct processes and can be implemented by different systems:

Authentication is the process of verifying who a "user" is.

Authorization is the process of providing a "user" access to a specific resource.

Note that the term user is used broadly in the context above. It may refer to an actual end-user, i.e., an application developer, a CECCM component, a microservice, an API client, etc.

4.3.4.1 NetScaler Authentication and Authorization Offload

Authentication and Authorization requirements are typically quite complex to implement and enforce. Towards this end, complex infrastructures introduce an authentication and authorization layer in front of applications and components that need to be protected. This layer can integrate with a variety of authentication platforms, establishing the user identity. Once the client identity is established, the same layer can enforce the appropriate policies.

A typical NetScaler (also known as Citrix Application Delivery Controller or Citrix ADC) deployment providing authentication controls is illustrated in the image below. In this particular example, the Security Assertion

Markup Language (SAML) protocol is used to authenticate a user against an Identity Provider. Once the user identity is established, he can access protected resources through the NetScaler.

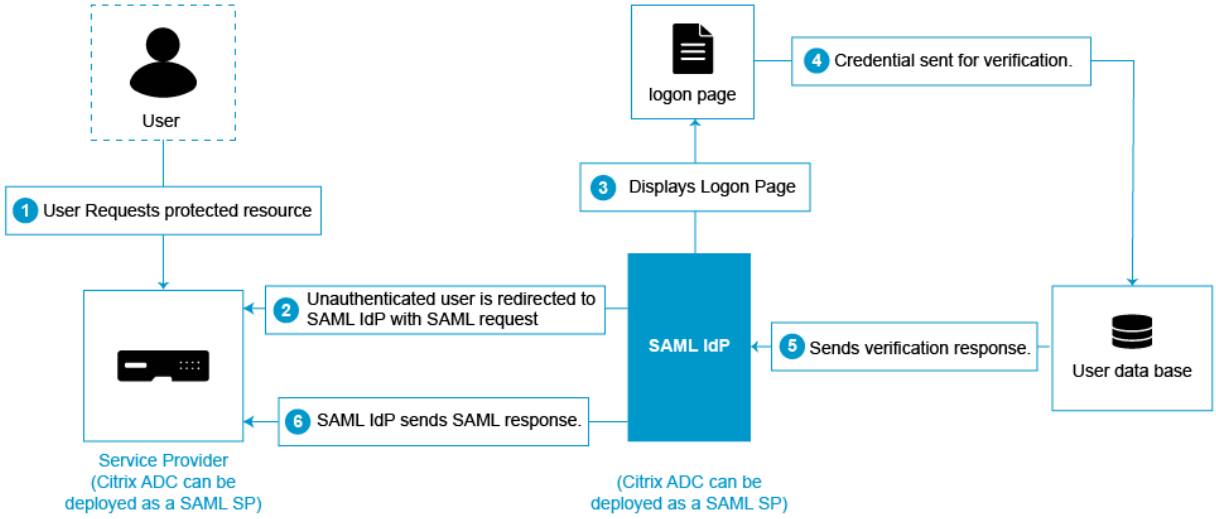

Figure 10. NetScaler Integration with SAML IdP

Offloading authentication and authorization are critical, since the respective landscape is simply too complex for an application and service admin to handle on his own. For example, NetScaler supports an exhaustive list of authentication protocols and methods, including but not limited to LDAP, RADIUS, SAML, OIDC, Smart cards, Client certificate authentication and reCAPTCHA. It provides a number of second factor authentication methods, such as email OTP, SMS OTP and others. Last, but not least, it has been verified to work with a number of leading industry vendors, such as Microsoft Active Directory (MAD) and Azure Active Directory (AAD), Okta, Ping Identity, Symantec and others.

4.3.4.2 NetScaler Authentication and Authorization

4.3.4.2.1 SAML

NetScaler provides extensive support for several authentication protocols and methods. However, the most common one for large infrastructures and fitting the needs of the AC^3 project is SAML. A typical SAML deployment is illustrated in Figure 10 above, with NetScaler deployed as a SAML Service Provider. The typical benefits of SAML are the following:

Single Sign On: The user needs to sign-on once against the SAML Identity Provider (IdP) with his credentials. Once the sign-on is complete, the same SAML token can be reused across multiple applications so as to not require the user to resend his credentials.

Reduced Risk of Credentials Theft: Authentication is performed directly against the SAML Identity Provider, rather than NetScaler. There is no need to synchronize user databases, hence there is a reduced risk of credentials theft.

Improved Security of Identity Provider: Since the NetScaler intercepts all traffic to the SAML IdP, it eliminates the need to directly expose it to the internet or otherwise malicious consumers.

SAML Validation: SAML assertions can be validated, even in the presence of a small-time skew, which is somewhat common between the NetScaler SP and the IdP.

Authorization: NetScaler can decrypt SAML assertions and extract contained attributes and values, including multi-valued ones. Up to 40Kbytes of attributes and values may be extracted from a SAML assertion. These values can subsequently be used for authorization control against arbitrary resources, applications and services, as long as these services are fronted by NetScaler.

4.3.4.2.2 *Two Factor Authentication*

Two-Factor Authentication, commonly abbreviated as 2FA, is typically implemented to better protect both user's credentials and the resources the user can access. 2FA methods rely on a user providing a password as the first factor and a second, different factor -- usually either a security token or a biometric factor, such as a fingerprint or facial scan.

NetScaler, thanks to its support for n-Factor Authentication allows easy enablement of additional factors. This is especially critical in case the IdP only provides support for single-factor authentication. Additional factors supported by NetScaler include Native One Time Password (OTP), Email OTP, Push OTP, SMS OTP, and FIDO2 for passwordless authentication.

4.3.4.2.3 *Auditing*

Using NetScaler as a single point of entry for authentication and authorization can greatly improve the auditing security posture of CECCM. NetScaler offers fine-grained auditing policy capabilities that allow selective auditing events based on the following:

- Applications accessed;
- Users or user groups being authenticated for access;
- Successful or failed authentication and authorization events (or both).

Depending on the desired posture and auditing or regulatory requirements all authentication and authorization events may be logged. In addition, said events may be exported to 3rd party Security Information and Event Management (SIEM) solutions deployed in the CECC, such as an ELK (Elasticsearch, Logstash, Kibana) stack.

4.3.4.2.4 *Microservices Specific Capabilities*

When considering CECCM microservices there are two types of traffic patterns typically encountered:

North-South Traffic: this corresponds to ingress traffic from external stakeholders, i.e. application developers, API clients, CECC infrastructure and others.

East-West Traffic: this corresponds to traffic within the CECCM, i.e. traffic between microservices implementing the CECCM.

When considering security for North-South traffic, all the authentication and authorization capabilities outlined in the previous sections are applicable. When considering security for East-West traffic, NetScaler can be deployed as a microservice sidecar, as illustrated in Figure 11.

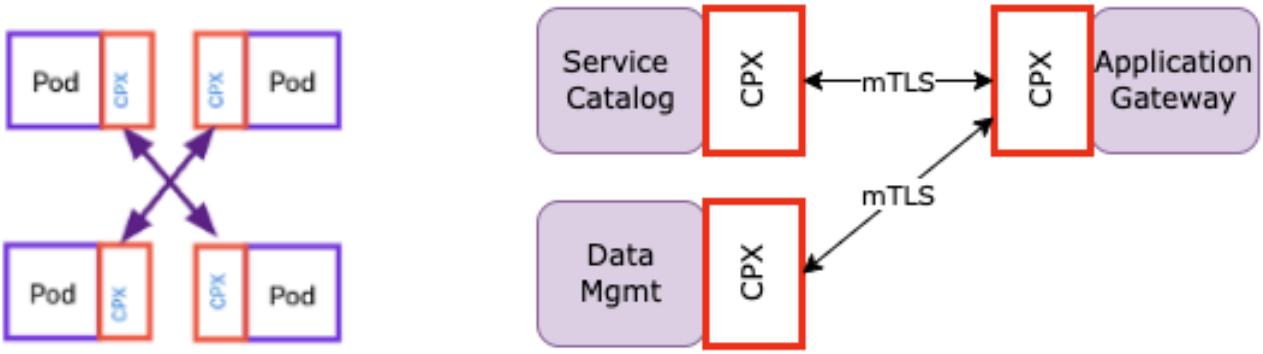

Figure 11. NetScaler CPX Service Mesh

When deploying NetScaler as a microservice sidecar, it can transparently perform mutual TLS (mTLS) between microservices, and hence enhance the authentication and authorization posture of the deployed microservices, without the application developer having to worry about application updates. Note that while the Figure above only lists three CECCM microservices, this is intended for purely illustrative purposes.

NetScaler can be used as a sidecar to accommodate any microservice-to-microservice communication to provide authentication offload capabilities.

4.3.5 Application Security

NetScaler provides advanced application security[6] with Web App Firewall (WAF)[7]. It offers robust defense, shielding production applications from potential threats and ensuring the integrity of sensitive business and client information. This advanced security solution is designed to prevent security breaches, data loss, and guard against unauthorized modifications to websites.

NetScaler's application security capabilities may be optionally leveraged to provide additional protection for any service communication. They are typically enabled in cases where trust boundaries are crossed and service trust cannot be established, such as the CECCM to Federated Infrastructure service communication outlined in Subsection 4.2.3.

The main NetScaler WAF features that will be considered in the context of AC^3 are the following:

Advanced Filtering: NetScaler WAF employs smart filtering of both requests and responses. Through this process, every data exchange is scrutinized for evidence of malicious activity.

Dynamic Threat Detection: NetScaler goes beyond traditional security measures by dynamically identifying and blocking not only common types of attacks but also emerging, yet-to-be-identified threats.

Vulnerability Safeguard: Beyond access protection, NetScaler fortifies against vulnerabilities in various layers. Whether it's legacy CGI code, scripts, web frameworks, or underlying operating systems, the Web App Firewall acts as a comprehensive shield against potential exploits.

Improved Encryption: Advanced TLS termination features, including support for TLS 1.3, HTTP Strict Transport Security (HSTS), OCSP stapling for efficient certificate revocation, as well as TLS session reuse and ticket mechanisms, serve to minimize the performance impact of TLS encryption without compromising security.

4.3.5.1 WAF Flow

Figure 12 illustrates a Flowchart of Web App Firewall Filtering. As presented in the diagram, when a user initiates a URL request on a protected website, the WAF initially examines the request to verify that it is not matching with any of the signatures. In the case of a signature match, the NetScaler WAF has the option to either display the error object or forward the request to the specified error page.

[6]https://docs.netscaler.com/en-us/citrix-adc/current-release/application-firewall/introduction-to-citrix-web-app-firewall.html

[7]https://www.netscaler.com/solutions/secure-application-delivery

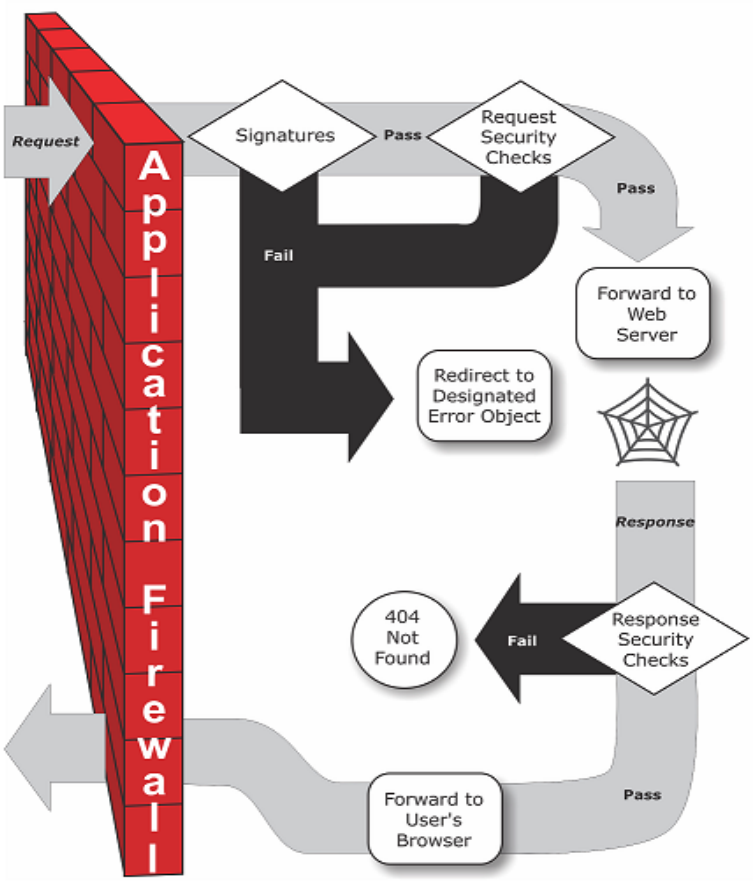

Figure 12. NetScaler Web Application Firewall Filtering (Decision Flowchart)

4.3.5.2 *WAF Monitoring*

During an operational state of application security, a transaction log is systematically generated in a structured format following any initiated security action. These transaction logs can be integrated into the broader SIEM, allowing for centralized monitoring and control of security policies. Note that NetScaler already includes support, though the NetScaler observability exporter, to seamlessly integrate with SLA and Security management modules such as Elasticsearch and Prometheus that will be used in AC^3 (see section 5.2), exporting the appropriate structured transaction logs and data streams.

4.3.5.3 *Rate Limiting and DDoS Protection*

NetScaler provides features for rate limiting[8] and DDoS protection[9] to help defend against various types of attacks and ensure the availability of applications. More specifically, it protects against DDoS attacks by implementing rate-limiting, as illustrated and described below.

The NetScaler enables the application or service administrator to:

- controls the number of requests each client can make to a particular service within a configured period;
- mitigate the impact of brute-force attacks, credential stuffing, and other forms of abuse;
- defend against large-scale, distributed attacks that aim to overwhelm a network or application;
- maintain service availability during DDoS attacks, preventing disruption and downtime.

[8] https://www.citrix.com/blogs/2020/09/29/protect-your-apps-with-rate-limiting-in-citrix-adc/
[9] https://www.netscaler.com/platform/ddos-protection

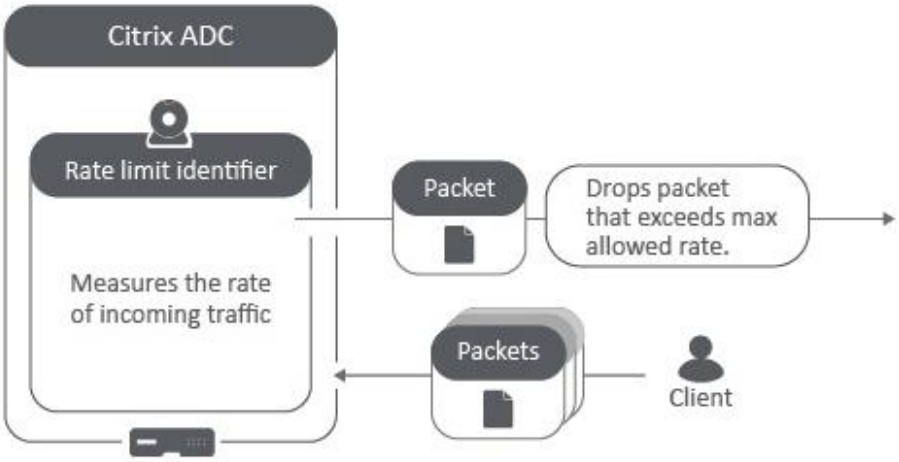

Figure 13. NetScaler WAF Rate Limiting Overview

4.4 Security of Data Management

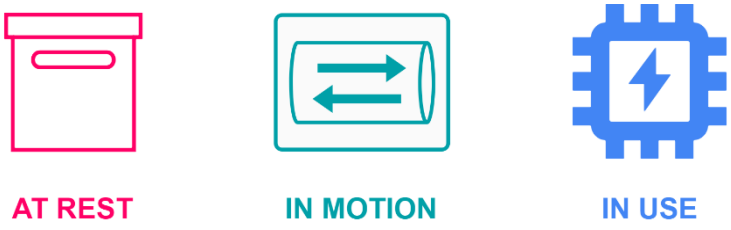

Figure 14. The (3) States of Data

Developing a secure application involves implementing numerous protective measures, with the utmost significance placed on those that ensure the security of the application's data. These security measures related to data are also the most challenging to put into practice. When it comes to safeguarding application data, two distinct categories (refer to Figure 14) necessitate protection: data in motion and data at rest. While in use data is safeguarded by the security measures implemented in the AC³ applications. **Data in motion** refers to data that is actively being transported from one place to another, whether it is over the internet or within a private network. Ensuring the security of data during its journey from one network to another or during the transfer from a local storage device to a cloud storage device is referred to as data protection in transit. Regardless of where data is on the move, it's essential to implement effective data protection measures for data in transit, as it is typically considered less secure while in the process of being transferred. **Data at rest** refers to data that is currently not in motion, meaning it is not actively being transferred between devices or networks. This includes data stored on devices like hard drives, laptops, flash drives, or data that is archived or stored in some other way. Data protection at rest focuses on securing this dormant data, regardless of the device or network where it resides. Although data at rest is sometimes perceived as less vulnerable compared to data in transit, attackers often consider it a more attractive target than data in motion. The level of risk associated with data in transit or data at rest is contingent on the security measures in place to protect data in either state.

4.4.1 In Motion Data Security

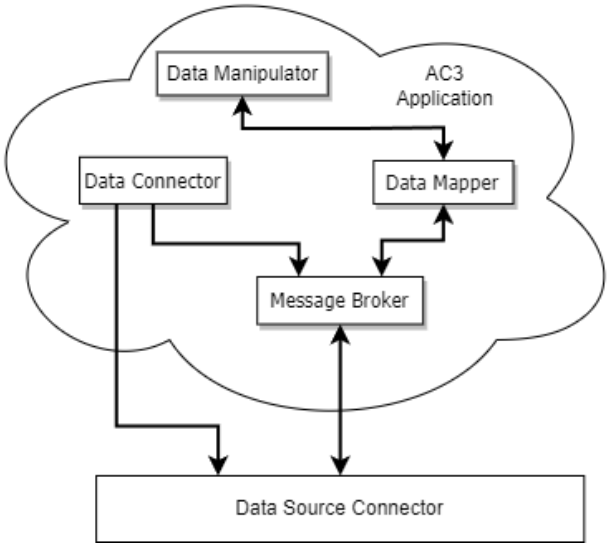

Figure 15. The Locations where Data Reside are Transferred and Processed in AC³

4.4.1.1 RabbitMQ

RabbitMQ[10] is a crucial part of the AC³ DM PaaS that is used to transfer and exchange data between the AC³ Data Sources and the data processing applications that consume them as part of the deployed AC³ applications (Figure 15). The data of AC³ are exchanged as messages sent through the RabbitMQ based Edge and Cloud brokers.

To safeguard sensitive information and ensure the secure transmission of data between messaging endpoints such as consumers and producers in RabbitMQ, SSL protocol is employed. SSL allows us to establish encrypted connections that prevent data compromise during transmission. RabbitMQ conveniently offers built-in SSL support, simplifying the implementation of this security measure. This support allows you to easily configure and enable SSL to enhance the confidentiality and integrity of data as it moves between different components of your messaging system, thereby bolstering the overall security of your RabbitMQ-based applications. The required certificates need to be generated and distributed accordingly to clients and servers of the AC³ DM PaaS.

Client authentication is another critical aspect. It validates the legitimacy of users or applications that seek access. Robust password hashing and salting techniques, combined with authentication mechanisms like SCRAM-SHA-256 or PBKDF2, stand against unauthorized access attempts.

To further fine-tune security, Role-Based Access Control (RBAC) comes into play. RBAC provides the control needed for access to RabbitMQ resources and actions. It operates by creating distinct roles, each associated with specific permissions and authorizations, and then assigns these roles to users or clients based on their functional requirements. The RabbitMQ configuration is adjusted by AC³ DM PaaS to enforce RBAC, ensuring that users or client applications can only perform actions aligned with their assigned roles, thereby enhancing security and control.

4.4.2 At Rest Data Security

Data in AC³ may need to be stored at some points of the DM PaaS until processing is possible and completed successfully. To secure this data, the AC³ DM PaaS uses suitable encryption mechanisms so that access to the physical infrastructures does allow for unauthorized access to the actual data stored in each location. In the

[10] https://www.rabbitmq.com/

rest of this subsection, we will refer to the tools we use to secure and encrypt AC³ data at rest in the different AC³ locations (edge or cloud).

4.4.2.1 *Secure Databases*

Securing databases at the edge and cloud is an essential practice to protect data stored on AC³. At the core of securing data at rest lies encryption of the databases used to store them. This fundamental measure employs encryption mechanisms like the Advanced Encryption Standard (AES) to encrypt the database files, safeguarding data from unauthorized access. This entails generating encryption keys and specifying encryption settings. The careful management of encryption keys is of paramount importance to ensure the secure storage and handling of these keys. Access control is another critical component of securing databases. It serves to restrict access to the database solely to authorized processes and users. Robust access control mechanisms, including Role-Based Access Control (RBAC) or Discretionary Access Control (DAC), are employed to define who may access the database. Authentication methods, such as OAuth2 or Multi-Factor Authentication (MFA), are utilized to verify user access. To implement access control, user roles and permissions need to be defined within the database system and user authentication needs to be configured. This ensures that only authorized users and processes can interact with the database, maintaining security and data integrity. Data retention policies are also systematically implemented to automatically delete data based once used by the process that requested them. This serves to enhance data security by minimizing data exposure and diminishing the risk of sensitive data lingering in the database beyond its intended use.

The Secure Database for IONOS Cloud is a comprehensive and robust database solution designed to provide a high level of security and data protection for businesses and organizations using IONOS Cloud services. It offers several key security features and capabilities to ensure the confidentiality, integrity, and availability of the data stored in the database. Some of the primary security features of the Secure Database for IONOS Cloud include:

Encryption at Rest and in Transit: The database implements encryption mechanisms to secure data both at rest, stored on disk, and in transit, while it is being transferred over networks. This ensures that sensitive information is safeguarded from unauthorized access and interception.

Access Control and Authentication: The Secure Database for IONOS Cloud enforces strong access control measures, employing role-based access control (RBAC), multi-factor authentication, and strict authentication protocols to manage user permissions and prevent unauthorized access to the database.

Compliance and Certifications: The database solution is designed to comply with industry standards and regulations, such as GDPR, HIPAA, and PCI DSS, C5 type1 to ensure that data secure handling and storage align with legal and regulatory requirements.

Automated Backup and Disaster Recovery: It includes integrated backup and disaster recovery capabilities to protect against data loss and facilitate rapid recovery in the event of unforeseen incidents, such as system failures or accidental data deletion.

Monitoring and Security Auditing: The Secure Database for IONOS Cloud incorporates monitoring and security auditing functionalities to track database activities, detect anomalies, and generate audit logs for compliance and security analysis.

Regular Security Updates: The database solution is regularly updated with the latest security patches and software updates to mitigate vulnerabilities and protect against potential security threats.

These security features collectively contribute to the overall integrity and confidentiality of data stored in the Secure Database for IONOS Cloud, providing organizations with a secure and reliable platform for their database management needs.

4.4.2.2 *IonosS3 Storage Encryption*

IONOS S3 storage encryption refers to the capability of securing data stored in IONOS Cloud's object storage

service, S3 (Simple Storage Service), using encryption techniques to protect the confidentiality and integrity of the stored data. This encryption ensures that the data is protected from unauthorized access and maintains its security throughout its lifecycle within the S3 storage environment. The key aspects of IONOS S3 storage encryption include:

1. **Data at Rest Encryption:** IONOS S3 storage encryption allows data to be encrypted while it is stored on the S3 servers. This means that even if someone gains unauthorized access to the physical storage media, the encrypted data remains unintelligible and protected.
2. **Server-Side Encryption:** IONOS S3 supports server-side encryption options, which means that the encryption and decryption processes are handled by the S3 service itself, providing a seamless and secure approach to data storage.
3. **Encryption Algorithms:** IONOS S3 storage encryption utilizes strong and industry-standard encryption algorithms to protect the data, ensuring that it remains secure and tamper-proof while stored in the S3 environment.
4. **Client-Side Encryption:** IONOS S3 also provides support for client-side encryption, allowing customers to encrypt their data on the client side before transmitting it to S3. This gives customers full control over the encryption process and the keys used to protect their data.

By offering robust encryption capabilities, IONOS S3 storage encryption provides organizations with the means to secure their data in the cloud, ensuring that sensitive information remains confidential and protected from unauthorized access. This is particularly crucial for complying with industry regulations and maintaining the trust and privacy of customer data.

5 Trust in AC³

To improve trust among the different stakeholders, which provide CECC infrastructure, AC³ proposes leveraging the overall architecture with trust management functionalities. The envisioned trust management procedures consist of building trust profiles for the different infrastructure providers involved in the federation. These profiles indicate how much the infrastructure provider is trusted in terms of supporting and validating the SLA established between the CECCM and the infrastructure providers, as well as with the application developer. Indeed, as introduced in D2.1, the application developer describes the application components, including information on the images, micro-service configuration, and, most importantly, the SLA. Generally, the SLA includes parameters on the expected performance of the micro-services. In AC³, we consider parameters such as the expected availability of the overall service, the expected link capacity and latency between the different micro-services, etc. The CECCM considers this SLA when deploying the micro-service components of the application on the CECC. This process consists of selecting the infrastructure providers for deploying the different micro-services. In AC³, the Resource Broker, a component of the CECCM, selects the infrastructure providers from the federation to deploy the micro-service components. Usually, the resource broker selects the infrastructure providers according to available resources exposed by the infrastructure provider and the cost of the resources. In AC³, we will add the infrastructure provider's reputation as a criterion for selecting the resource provider from the federation. We recall that one of the key innovations of the project is the separation of resources from application management. The CECCM owner can run and deploy applications on the CECC infrastructure without owning an infrastructure. The main approach is to use a federation of resources.

5.1 Trust Model

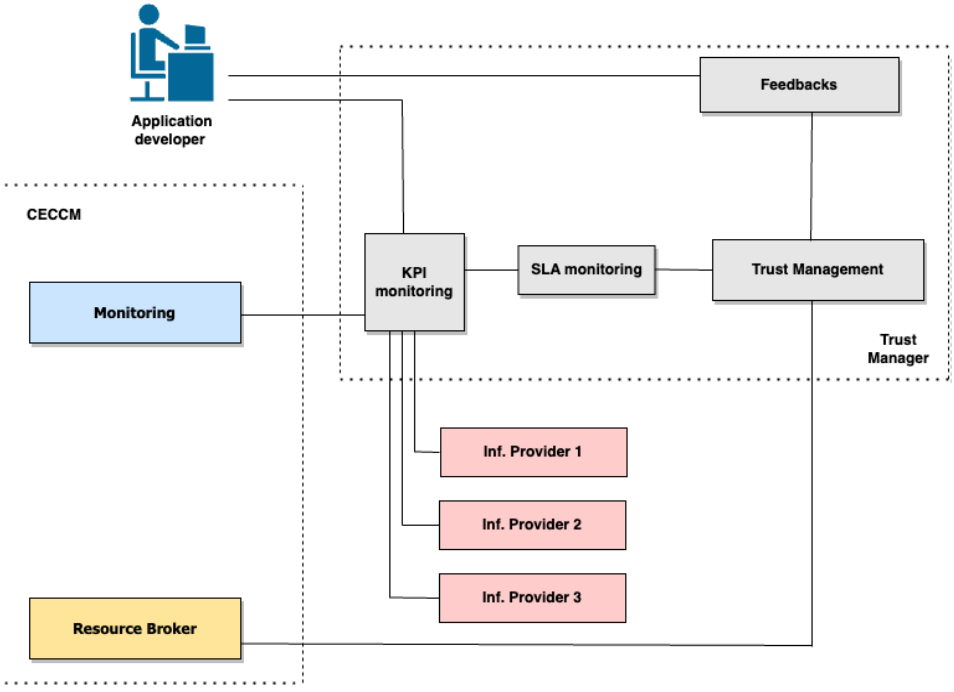

Figure 16. Trust Model Overview

To build the infrastructure provider's reputation, we devise a novel trust architecture, as shown in Figure 16. Besides the already defined components of the CECCM (i.e., Monitoring and Resource Broker), the trust architecture model introduces a new component or entity named Trust Manager. The latter is an independent entity that belongs to the federation manager (refer to D2.1). It aims to derive the reputation of each infrastructure provider. The trust manager is composed of:

1. **The KPI Monitoring Module:** It collects monitoring data regarding the SLA performances from the

different involved actors, i.e., CECCM, infrastructure providers, and the application developer. The collected data is gathered from the different actors to detect which entity is respecting or violating the SLA. The monitoring data collected from the application developer will indicate if the application sees a violation of the SLA. If so, the collected data from the CECCM and infrastructure providers will allow for the detection of which entity is violating the SLA. Indeed, in AC^3, micro-service components of the same application can be deployed on different infrastructure providers. So, it is important to have access to the monitoring information from the perspective of CECCM and infrastructure providers.

2. **The SLA Management Module:** it aims to exploit the collected monitoring data to detect SLA violations automatically. To this end, we will explore the usage of Smart Contracts to establish SLA and use monitoring information to detect violations.
3. **Feedback Module:** it collects the feedback of the end user (i.e., application developer) about the service and the user experiences. Generally, it is collected periodically or after the end of the application.
4. **The Trust Management Module:** uses the output of the SLA monitoring module and the Feedback module to derive the reputation value of the different infrastructure providers. The reputation will be published on a blockchain. Then, the reputation of an infrastructure provider will be updated. The reputation value will be used later by the resource broker when selecting resource providers from the federation.

5.2 SLA Management Using Smart Contracts

In the AC^3 project, the management and monitoring of SLA is critical to build the reputation of the different stakeholders involved in deploying the micro-service-based application over the CECC infrastructure. To this aim, in AC^{3,} we envision the utilization of Smart Contracts to manage SLAs effectively. Relying on this approach, the SLA management process is automated to detect SLA violations. In what follows, we introduce some principles and go deeper into the SLA management process.

5.2.1 SLA Definition

Within the AC^3 context, SLAs meticulously define parameters governing the expected performance of micro-services. These parameters encompass critical aspects such as service availability, link capacity, and latency between micro-services. Notably, this information is obtained during the application development phase when interacting with the OSR module of the CECCM. The SLA acts as a crucial document that establishes a comprehensive framework, ensuring the reliability, availability, and performance of the service. It serves as a foundation for collaboration between the service and its underlying infrastructure throw the CECCM.

The SLA initiation process involves specifying key service metrics to evaluate performance, including maximum response time, optimal uptime, and availability. Additionally, the SLA sets clear expectations for throughput requirements, providing a solid foundation for assessing the overall success of the service. Scalability is addressed through explicit limits and guidelines, detailing how the infrastructure should dynamically scale to handle increased demand throughout the AC^3 project.

To enhance service reliability, the SLA defines acceptable error rates and redundancy requirements. This strategic approach aims to mitigate the impact of hardware failures or disruptions, ultimately enhancing the overall stability and resilience of the service. Security considerations are integral, with detailed provisions for access controls, authentication protocols, and encryption standards, aligning with best practices for safeguarding sensitive information within the AC^3 project.

Provisions for data integrity and recovery are outlined, specifying the frequency of backups, Recovery Time Objectives (RTO), and procedures to restore service in the event of a failure. These measures minimize downtime and ensure consistent availability of data and services. Monitoring and reporting mechanisms are integral components, identifying tools and methodologies for monitoring infrastructure performance, coupled with reporting requirements to keep stakeholders informed.

Clear communication is emphasized in the SLA, specifying communication channels and notification procedures. This ensures that stakeholders are promptly informed of any service outages, maintenance activities, or critical events, fostering transparency and accountability throughout the AC^3 project.

In summary, the SLA serves as a comprehensive technical agreement, aligning service and infrastructure components within the AC^3 project. By addressing key metrics, scalability, reliability, security, backup and recovery, monitoring, incident response, compliance, and communication, the SLA establishes a robust foundation for delivering a reliable, secure, and high-performing solution.

5.2.1.1 SLA Structure

In a manner akin to any contractual agreement, SLA is a meticulously organized collection of elements, encompassing both mandatory and optional components. These elements delineate various aspects, including the duration of validity, the involved contracting parties, the type of services provided, guarantees related to objectives, penalties, clauses for suspension or termination, and compensation. Figure 17 shows the SLA structure including the following components:

- **Period of Validity:** As illustrated in Figure 17, the timeframe of an SLA is determined by the initiation and conclusion dates of the network resource lifecycle. This period can be altered in accordance with stipulated termination conditions.
- **Parties Involved**: This section outlines the entities engaged in the contractual agreement, namely the three signatory parties – Application Developer (AD), CECCM, and CECC Infrastructure Providers (InfPs). Additionally, it introduces the trusted fourth party, the trust monitoring system. The latter is instrumental in validating the proper functioning of the desired service, assessing its availability, or determining if it falls below the agreed-upon performance level stated in the SLA. This segment provides details about the parties, including their names and contact information.
- **Template:** An integral component of the SLA, the template defines parameters ensuring the realism and achievability of the offered QoS. This vital section encompasses five key elements: service type, parameters, guarantees, billing, and termination conditions.
- **Service Types:** It facilitates the diversity of services offered without interferences between the services. Each service's priority variation necessitates distinct SLAs, considering different services and operational modes.
- **Parameters:** This section defines variables utilized in other parts of the contract, detailing specific elements such as metrics and monitoring types. Metrics are identified by an identifier, a description (e.g., latency, throughput, reliability), and a corresponding unit (e.g., ms, bps, %). Monitoring specifies the nature of the evaluated metric, including aspects such as average, maximum, minimum, evaluation window, and frequency of value collection.
- **Guarantees:** This segment encompasses three key elements – requirements, terms, and penalties.
- **Requirements:** Essential specifications for the service's optimal functionality are presented here. Specifications may be mandatory or optional, detailing preliminary technical elements to be fulfilled.
- **Terms of Guarantees:** Detailing the Service Level Objectives (SLO) of the contract, this part elaborates guarantees through terms (e.g., reliability, latency, bandwidth, throughput) and objectives using operators like "Or" and "And."
- **Penalties:** In the event of a degradation in the end-to-end network performance (SLA violation), CECCM is obligated to compensate AD. Simultaneously, InfPs involved in the SLA violation due to resource degradation must compensate the CECCM. The amount of reimbursement is mutually agreed upon and described in the penalties section.
- **Billing:** Billing generally follows two models – "pay package" or "pay as you go." In this scenario, the "pay as you go" model is adopted, where the service price depends on operating modes and penalties.

- **Termination:** A contract can terminate automatically at the SLA end date, with termination types including voluntary termination at the discretion of the AD, termination due to violation detected by the monitoring system, and termination at the end of the SLA lifecycle.

By structuring the SLA in this manner, it ensures clarity, comprehensiveness, and adaptability to the dynamic requirements of AC³ framework.

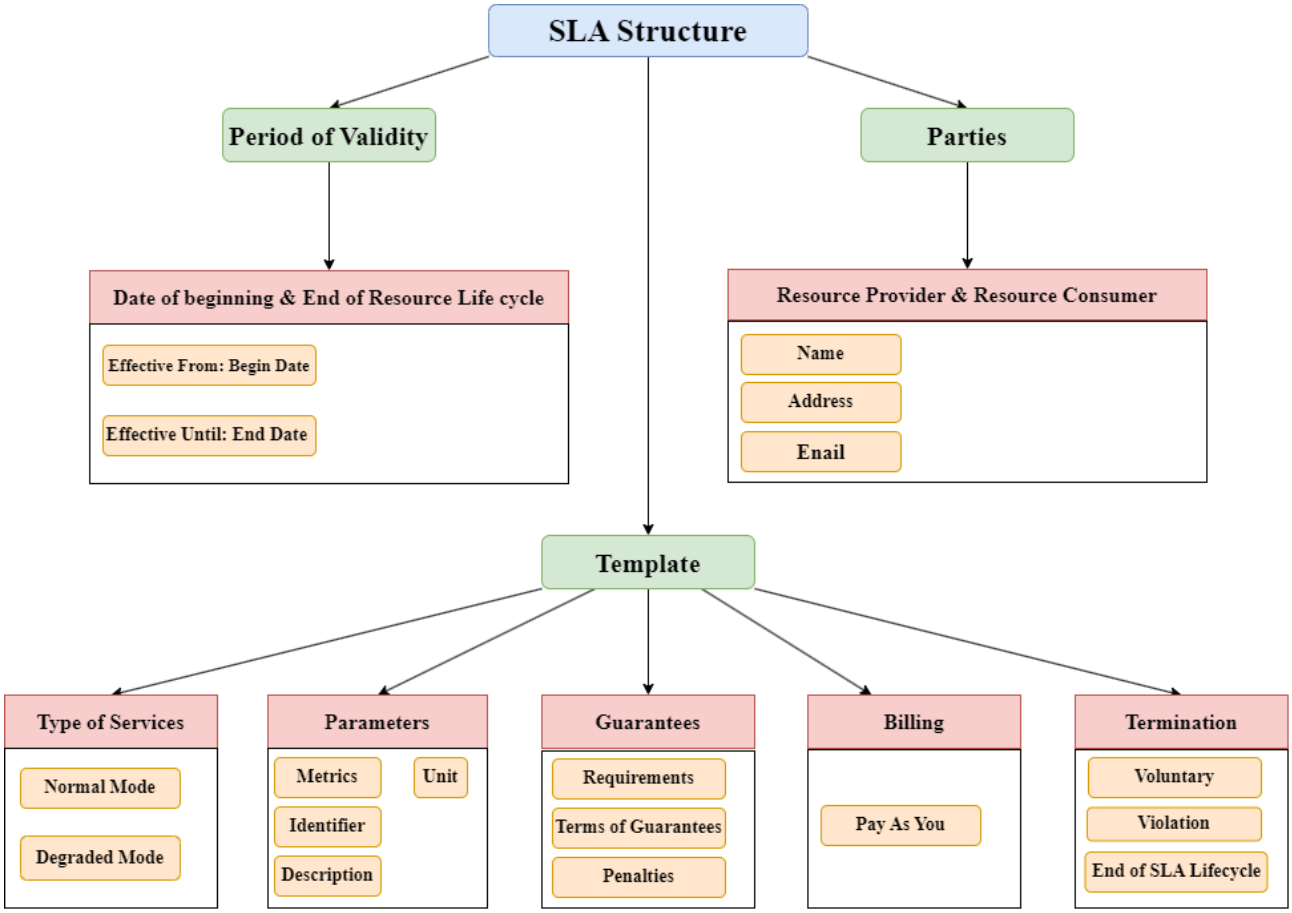

Figure 17. Service Level Agreement Structure

5.2.2 Smart Contracts Definition and Blockchain Integration

In the AC³ project, Smart Contracts serve as self-executing contracts with predefined terms directly encoded into code. Specifically, within SLA Management, these programmable scripts enforce SLA terms and conditions on a decentralized network. Operating on principles of transparency, security, and automation, Smart Contracts execute predetermined actions, such as addressing SLA violations, within a decentralized network environment.

Expanding beyond SLA Management, Blockchain, a distributed ledger system, proves instrumental for securely recording transactions. Utilizing cryptographic codes like "hash" through algorithms such as "Proof of Work," Blockchain ensures secure verification of transactions by multiple nodes. Blockchain platforms encompass permissioned, permissionless (public), and consortium types, each chosen based on factors like network privacy and access control. The application of Blockchain extends across diverse domains, including healthcare, and IoT demonstrating a high versatility. Complementing Blockchain, Smart Contracts automate transactions within the Blockchain network, eliminating the need for intermediaries and ensuring transparency and security.

Within the AC³ project, Blockchain and Smart Contracts offer potent tools for decentralized application scenarios. While various works advocate for Blockchain and Smart Contracts in contexts like 5G networks [5]

[6], the AC³ project tailors these concepts to its specific needs. The focus lies on enhancing transparency, security, and automation in operational scenarios, catering to the unique requirements and challenges of the project.

5.2.3 SLA Establishment and Management

In the AC³ project, the establishment and management of Service Level Agreements (SLAs) play a pivotal role in ensuring seamless interaction between **CECCM** and the **InfPs**, as well as with the **AD**. This process initiates with the AD Micro-Service (MS) selecting a Quality-of-Service (QoS) aligned with its specific requirements. The MS requests the creation of a new SLA contract throw the CECCM, using a predefined blueprint or template provided by the InfP.

Subsequently, the InfP translates this template into detailed resource requirements for each MS, encompassing computing resources, network resources, radio resources, and more. The deployment of the MS employs a resource broker mechanism to determine the suitable InfP for deployment. The Resource Broker utilizes a proprietary algorithm considering various criteria, including the availability of resources and their cost, but also the reputation cost, to select infrastructure providers from the federation to deploy micro-service components.

Once resources for each microservice are allocated, indicating readiness for end-to-end deployment, SLA contracts are executed between the AD and the InfPs. The SLA serves as a comprehensive agreement specifying expectations and commitments. It is instrumental in verifying if the defined service is delivered as contracted, and SLA management aids in addressing QoS degradation, outlining requirements such as bandwidth, throughput, and latency.

5.2.4 Trust Architecture Utilizing Blockchain and Smart Contracts

To enhance security, anonymity, and automation in negotiating and managing SLAs, AC³ proposes a trust architecture utilizing Blockchain and Smart Contracts (See Figure 18). The architecture facilitates the negotiation and deployment of resources, managing SLAs among all involved entities. The proposed trust architecture model in Figure 16 incorporates a resource broker, used by the CECCM, selecting the appropriate InfP for deploying a Micro-Service based mostly on the reputation of each InfP.

The framework enables the monitoring of KPIs for automatically detecting SLA violations, with the reputation of InfP dynamically adjusted based on SLA compliance. This pioneering work contributes a trust architecture for the negotiation and deployment of resources, addressing the complexities of managing multiple SLAs involving diverse stakeholders. Unlike previous approaches, the proposed framework combines Blockchain-based brokering mechanisms with a reputation-based selection algorithm, offering a more nuanced and comprehensive solution.

In the AC³ project, the establishment of an end-to-end network virtual or physical resources across different InfPs is paramount. This involves negotiations for necessary resources across various CECC providers, ensuring trust between the infrastructure providers and the micro-services components.

AC³ trust architecture, as depicted in Figure 16, facilitates the negotiation and deployment of an end-to-end network resources in the AC³ project. This architecture involves key entities AD, CECCM, and CECC. AD utilizes the Blockchain layer for contract generation and negotiation, with the resource broker overseeing the selection of InfP for each MS component based on the KPI Monitoring from the CECCM and the CECC but also the Feedback from the AD. The end-to-end resources are ready for deployment upon the successful negotiation and finalization of all the MS components contracts.

The subsequent phases cover resource brokering, deployment, InfP reputation, and the algorithm of Trust computation. These steps involve the generation of the Contracts, negotiation based on Blockchain, monitoring SLAs, updating InfPs reputation, and employing a resource broker algorithm for optimal InfP selection.

5.2.5 Tools and Frameworks

1. **Smart Contract Platform:** In the realm of AC^3, the choice of Ethereum[11] as the underlying blockchain platform for Smart Contracts is deliberate and strategic. Ethereum's decentralized and programmable nature perfectly aligns with the project's overarching requirements. Serving as the bedrock of the Smart Contract infrastructure, Ethereum provides a secure and transparent environment crucial for the execution and enforcement of Service Level Agreements (SLAs).
2. **KPI Monitoring Module:** Prometheus[12] is employed for real-time monitoring of Key Performance Indicators (KPIs). It collects and stores monitoring data related to SLA performances from entities such as CECCM, infrastructure providers, and application developers. It ensures a comprehensive and up-to-the-moment dataset crucial for informed decision-making within the SLA framework.
3. **SLA Monitoring Module:** At the heart of SLA enforcement lies the SLA Monitoring Module, empowered by the utilization of Solidity[13], a Smart Contract Programming Language. It facilitates the establishment of dynamic and programmable SLAs within the decentralized Ethereum network. Solidity provides the expressive syntax and functionality required for crafting Smart Contracts that define and enforce SLAs. This dynamic language facilitates the establishment of programmable SLAs within the decentralized Ethereum network, enabling real-time adaptability to changing conditions.
4. **Feedback Module:** The Feedback Module harnesses the capabilities of the ELK Stack[14], comprising Elasticsearch, Logstash, and Kibana, for efficient data management. This toolset is instrumental in the systematic collection, storage, and analysis of qualitative feedback from end-users. ELK Stack's prowess in handling data allows the project to gain profound insights into user experiences and satisfaction, forming a crucial component of the iterative SLA management process.
5. **Trust Management Module:** In the realm of trust and reputation, the Trust Management Module integrates the Hyperledger Fabric[15] blockchain framework. Hyperledger Fabric plays a pivotal role in ensuring the transparency of reputation values within the blockchain. This integration enhances trust and accountability by securely publishing reputation values, offering a decentralized and tamper-resistant ledger for tracking the credibility of infrastructure providers over time.

5.2.6 Workflow

Our system is designed with a comprehensive approach to ensure robust performance and trustworthiness. At the core of our architecture, Prometheus serves as the vigilant observer, continuously monitoring KPIs such as service availability, link capacity, and micro-service latency. This real-time monitoring provides essential data for assessing the system's health and performance.

Complementing the KPI monitoring, our system employs Smart Contracts written in Solidity for SLA definition and enforcement. These contracts not only articulate SLAs but also autonomously monitor adherence to the defined terms and conditions. Interacting seamlessly with the Ethereum blockchain, these Smart Contracts ensure that SLAs are not merely documented but actively enforced in real-world scenarios.

In parallel, we recognize the significance of end-user feedback in maintaining a dynamic and responsive system. The ELK Stack stands as a robust solution for the continuous collection of qualitative feedback from end-users. This feedback loop is essential for making informed adjustments and improvements based on real user experiences.

To tie these elements together and establish a reliable trust management system, we leverage Hyperledger Fabric integrated into the Trust Management Module. Here, reputation values are meticulously derived from both the outputs of SLA monitoring and the qualitative feedback gathered. The resulting reputation values

[11] Ethereum - https://ethereum.org/fr/developers/docs/smart-contracts
[12] Prometheus - https://prometheus.io/
[13] Solidity - https://docs.soliditylang.org/en/v0.8.24/
[14] ELK Stack - https://www.elastic.co/fr/elastic-stack
[15] Hyperledger Fabric - https://www.hyperledger.org/projects/fabric

are securely published into the blockchain, fostering transparency and accountability in our trust management approach. This interconnected system ensures that real-time monitoring, SLA enforcement, user feedback, and trust management work in tandem to uphold the integrity and reliability of our infrastructure.

To provide a comprehensive visualization of this interconnected system, refer to the accompanying Figure 18 that encapsulates the holistic architecture and workflow.

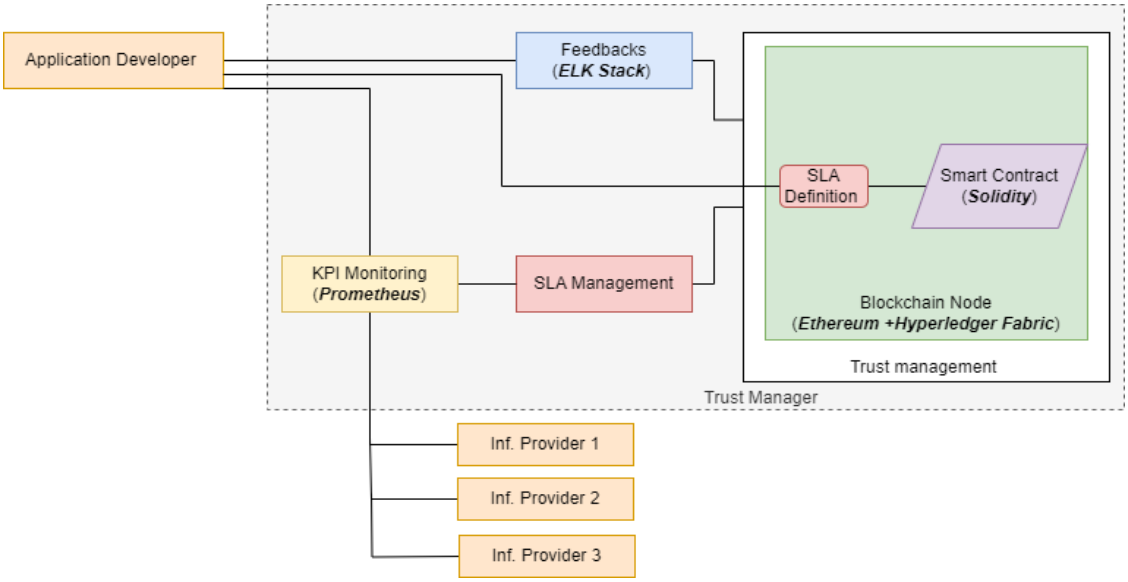

Figure 18. Trust Model (Tools and Frameworks)

5.3 Trust of Data Management

Dataspaces, integral to the AC^3 data management framework, aim to establish secure and sovereign environments for data exchange. This approach ensures that participants are fully aware of and in control of their data, with trust being a crucial aspect to uphold, especially within the Data Management PaaS. Thus, it is essential to understand the technical challenges and seek solutions to maintain this trust and control.

Originating over eighteen years ago within computer science, the concept of dataspaces represented a shift from traditional central databases towards storing data at their source [7]. This evolution was significantly advanced by Fraunhofer ISST[16], leading to the creation of the International Data Spaces Association (IDSA)[17] in 2015. This body was instrumental in setting the groundwork for dataspaces, as seen in the International Data Spaces Reference Architecture Model (IDS-RAM)[18], which outlines the critical components and their requirements. Furthermore, the Gaia-X[19] initiative, a European-led project, expanded dataspaces by promoting a federated and secure data exchange framework, aiming for interoperability among various cloud providers and IT infrastructures. Gaia-X's objectives are in line with enhancing data sovereignty and security, crucial for building trust in data exchanges.

The primary goal within the AC^3 framework, guided by the principles of IDSA and Gaia-X, is to confront and overcome the challenges in secure data management PaaS. Dataspaces are pivotal in this regard, facilitating data sharing even among competitors and addressing technological barriers to data sharing. The challenges are multifaceted, including the discovery of necessary data, secure publication, sharing in multi-cloud environments, and maintaining control over shared data to ensure its integrity and appropriate use. By

[16] https://www.isst.fraunhofer.de/
[17] https://internationaldataspaces.org/
[18] https://docs.internationaldataspaces.org/ids-knowledgebase/v/ids-ram-4/
[19] https://gaia-x.eu/

tackling these issues, the framework strives to promote a seamless, secure, and sovereign data exchange landscape.

According to the IDS-RAM, two basic types of Trust can be associated while establishing 'Trust' between the various participants for data sharing and exchange. The first category of Trust is 'Static Trust', which is dependent on the certification of the operational environment. The second category of 'Trust' is known as 'Dynamic Trust', which is based on active monitoring of the operational environment and core technical components[20].

According to the IDS-RAM, for making secure and trustworthy data sharing and exchange between participants, some preliminary actions are essential to be followed. Figure 19 presents how the different related components (e.g., Certification Authority, Dynamic Attribute Provisioning Service (DAPS), Participant Information Service (ParIS), Dynamic Trust Monitoring (DTM), etc.) interact with each other to ensure 'Trust' in data management plane of any computing ecosystem.

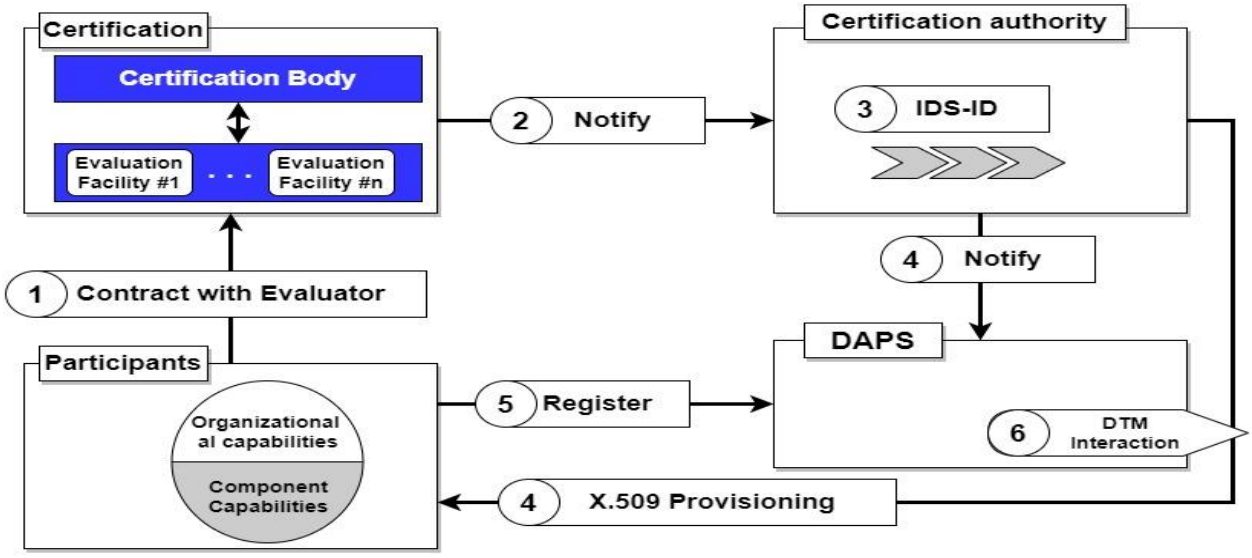

Figure 19. Interaction between different IDSA Components for Ensuring Trust in Data Management

Following the guidance of IDSA, the primary participant approaches an evaluation facility with a certification request, seeking to validate their operational environment and core components against IDS certification standards. Upon successful certification, the Certification Body informs the Certificate Authority (CA), which then acknowledges the validity of the certification. Following this, the Certificate Authority crafts a unique IDS-ID for the certified pair and issues a corresponding digital certificate. This certificate, specifically an X.509, is securely delivered to the participant and the DAPS is duly notified. With the X.509 certificate integrated into the component, it proceeds to register with the DAPS, enabling a DTM system to vigilantly track the component's behavior. Following this strategy and adhering to the overall mechanism, AC^3's data management PaaS will enable continuous and proactive safeguarding of the operational environment against potential security vulnerabilities or detecting attempted breaches.

However, the concept of the dataspace extends beyond the mere development of a Proof of Concept within the AC^3 framework. Our goal is not only to conceptualize dataspace but also to utilize it for sharing and exchanging data in pilot use-cases between service/data consumers and service/data owners. To achieve this, we have planned to embrace the Open-Source Eclipse Dataspace Components (EDC)[21]. This comprehensive framework encompasses the entire spectrum – from concept and architecture to code and

[20]https://docs.internationaldataspaces.org/ids-knowledgebase/v/ids-ram-4/layers-of-the-reference-architecture-model/3-layers-of-the-reference-architecture-model/3-1-business-layer/3_1_3_digital_identities
[21] https://github.com/eclipse-edc

samples, offering a fundamental set of features, both functional and non-functional, which can be adapted and personalized by dataspace implementations. Leveraging the framework's defined APIs ensures interconnectivity by design and customization options, guided by the specifications of the Gaia-X AISBL Trust Framework[22] and the IDSA dataspace protocol[23]. One significant factor driving our adherence to the EDC framework is the availability of the EDC_IONOS S3[24] extension. This extension seamlessly integrates with the Eclipse Dataspace Connector and enables operations with the IONOS S3 Storage. Notably, this storage solution is crafted with a comprehensive suite of security features, safeguarding critical databases and data against unauthorized access, breaches, or accidental data loss.

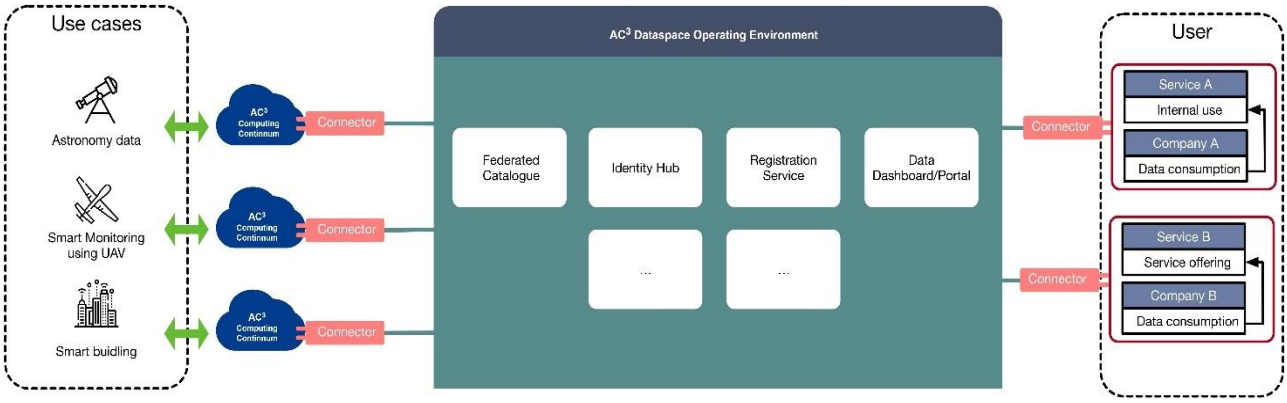

Figure 20. Conceptualized Vision of AC³ Dataspace Operating Environment

To simplify our testing plan and offer a clear visual of the overall implementation, we have introduced a conceptualized dataspace operating framework for AC³'s DM PaaS in Figure 20. This illustrative diagram presents the AC³ dataspace environment, featuring various components and tools, including the Federated Catalog, Trust services (e.g., Identity Hub, Registration services), and other tools. All these elements are thoughtfully aligned with the IDSA Guidelines and GFXS/XFSC, serving as the guiding principles in constructing the AC³ Federated Operating Environment. Notably, the diagram showcases the conceptualized dataspace framework for AC³, incorporating the participation of two cloud providers, IONOS and Arsys along with an Edge service provider from EURECOM. These computing continuum providers can host Kubernetes clusters, which can run numerous dataspace services and components. Within these Kubernetes clusters, Pods are deployed, each responsible for running Docker images containing the essential services of the dataspace, such as Connectors, Federated Catalog, other trust services, and data dashboard functionalities, ensuring a comprehensive dataspace environment.

[22] https://gaia-x.gitlab.io/policy-rules-committee/trust-framework/
[23] https://docs.internationaldataspaces.org/knowledge-base/dataspace-protocol
[24] https://github.com/ionos-cloud/edc-ionos-s3

6 Conclusion

This deliverable, titled "D2.5: Security and Trust Management for CECC," stands as a pivotal testament to the AC^3 project's commitment to redefining the landscape of federated computing through robust security and trust mechanisms. In the dynamic realm of modern digital interconnectedness, where seamless user experiences and real-time data processing are paramount, the integration of Cloud, Edge, and Far Edge technologies has become a necessity. The AC^3 project, embodied by the CECC framework and the innovative Cloud Edge Computing Continuum Manager (CECCM), addresses this need by fostering a federated computing continuum that seamlessly combines centralized cloud resources with the immediacy of edge devices.

At the heart of AC^3's vision lies the CECCM, an instrumental linchpin that underscores the criticality of resource federation across diverse landscapes, from centralized hubs to the farthest edges. This integration is meticulously designed to elevate resource availability while fortifying the system's overall resilience and adaptability through robust security and trust mechanisms. The CECCM, operating as a Platform as a Service (PaaS) for data management, facilitates seamless application development and deployment, embodying AC^3's user-centric philosophy.

This document delves into the multifaceted dimensions of security and trust management within the CECC ecosystem, providing novel contributions in forms strategies, methodologies, and mechanisms crafted to safeguard data, resources, and interactions across this diverse and dynamic landscape. As the AC^3 project progresses, this deliverable serves as an evolving blueprint, laying the foundation for forthcoming activities in WP3 and WP4. By M24, the final version of this deliverable, enriched by federated and open API integration, enhanced functional blocks, and feedback from ongoing project activities, will be unveiled. The AC^3 project, through its emphasis on security, trust, sustainability, and network programmability, is steadfast in forging a path toward a secure, resilient, and trustworthy Cloud Edge Computing Continuum.

7 References

[1] J. Kindervag, "No more chewy centers: The zero trust model of information security," *Forrester Research, Inc,* 23 March 2016.
[2] R. Ward and B. Betsy, "Beyondcorp: A new approach to enterprise security," 2014.
[3] YOUNG and S. D, "Moving the us government toward zero trust cybersecurity principles," 2021.
[4] "Qualys SSL Labs, SSL and TLS Deployment Best Practices, "Use Secure Protocols"," [Online]. Available: https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices.
[5] N. Boubakr, K. Adlen, H. Nicolas, F. Pantelis A. and M. Hassine, "A Blockchain-Based Network Slice Broker for 5G Services," in *IEEE Networking Letters*, 2019.
[6] S. J. Eder, R. B. Bruno, G. Z. Lisandro, and S. Burkhard, "Enabling Dynamic SLA Compensation Using Blockchain-based Smart Contracts," in *IEEE International Symposium on Integrated Network Management*, 2019.
[7] Halevy, M. Franklin and D. Maier, "Principles of dataspace systems," in The twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2006.