Grant Agreement N°: 101020259

Topic: SU-DS02-2020

ARCADIAN-IoT

Autonomous Trust, Security and Privacy Management Framework for IoT

D2.5: ARCADIAN-IoT architecture

Revision: v.1.0

Work package	WP 2
Task	Task 2.3
Due date	30/04/2021
Submission date	03/05/2021
Deliverable lead	IPN
Version	1.0
Partner(s) / Author(s)	IPN: Paulo Silva, Sérgio Figueiredo, João Rainho, Vitalina Holubenko, Gabriela Bento UWS: Jose M. Alcaraz Calero, Qi Wang, Antonio Matencio Escolar, Ignacio Sanchez Navarro, Pablo Benlloch Caballero, Ignacio Martinez Alpiste, Gelayol Golcarenarenji TRU: João Casal, José Rosa, Tomás Silva, Ivo Vilas Boas, Carlos Morgado XLAB: Tilen Marc, Benjamin Benčina, Jan Antić UC: Bruno Sousa MARTEL: Giacomo Inches BOX2M: Alexandru Gliga RISE: Alfonso Iacovazzi, Han Wang ATOS: Ross Little, Miguel Angel Mateo Montero RGB: Ricardo Ruiz LOAD: Pedro Colarejo

Abstract

This public report constitutes the deliverable D2.5 of ARCADIAN-IoT, a Horizon 2020 project with the **grant agreement number 101020259**, under the topic **SU-DS02-2020**. The main purpose of the report is to present the ARCADIAN-IoT framework, its planes and associated components. The material presented in this document is the main outcome of **Task 2.3 (ARCADIAN-IoT architecture specification)** and has considered inputs from Task 2.1 (Use cases specification and planning), Task 2.2 (Requirements for ARCADIAN-IoT framework) as well as initial research outcomes regarding individual components (addressed in WP3 and WP4). All technical partners were involved in the iterative process of architecture specification. Ultimately, this process led to the definition of the major functionalities of the framework, the characterisation of inter-component articulation and interfaces, as well as additional architectural information paving the path for future integration activities.

Keywords:

ARCADIAN-IoT Framework; Architecture Specification; Architectural views, Inter-component Articulation

Document Revision History

Version	Description of change	List of contributors
V0.1	Component Description	IPN, ATOS, LOAD, MAR, RGB, RISE, BOX2M, UWS, XLAB, TRU
V0.2	Methodology	IPN
V0.3	Architecture	IPN, ATOS, LOAD, MAR, RGB, RISE, BOX2M, UWS, XLAB, TRU
V0.4	Deployment and Development View	IPN (with contributions ATOS, LOAD, MAR, RGB, RISE, BOX2M, UWS, XLAB, TRU)
V0.5	Data Flows	IPN, ATOS, LOAD, MAR, RGB, RISE, BOX2M, UWS, XLAB, TRU
V0.6	Information and Operational Views	IPN (with contributions from all partners)
V0.7	Introduction, Abstract, Executive Summary review	IPN
V0.8	Operational and Concurrency Views	IPN
V0.9	General Corrections and review from ethical / legal point of view	All partners
V0.91	Deliverable review	TRU
V0.92	Deliverable updates according to TRU review	IPN
V0.93	Final internal review	IPN
V1.0	Final edits and document cleaning	IPN

Disclaimer

Project co-funded by the European Commission under SU-DS02-2020		
Nature of the deliverable:	R*	
Dissemination Level		
PU	Public, fully open, e.g., web	√
CI	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to ARCADIAN-IoT project and Commission Services	

* *R: Document, report (excluding the periodic and final reports)*
DEM: Demonstrator, pilot, prototype, plan designs
DEC: Websites, patents filing, press & media actions, videos, etc.
OTHER: Software, technical diagram, etc

EXECUTIVE SUMMARY

Deliverable 2.5 (ARCADIAN-IoT architecture) presents the ARCADIAN-IoT framework. It starts by introducing the concept, objectives and target Key Performance Indicators (KPIs) before providing a high-level description of the architecture and its planes. Its main components and associated interfaces (i.e., inter component interfaces) are then described.

Driven by previously defined use cases – while aiming to support additional ones -, and component-specific requirements and KPIs, the architecture was specified according to a Software Systems Architecture methodology. The main result is a holistic presentation of the ARCADIAN-IoT architecture, its building blocks, components and interdependencies among components. Moreover, initial characterization of related constraints such as target deployment locations, adopted development tools and methodologies are introduced.

As the architecture specification relates the research roadmap of the project with its innovative components and their major functionalities, it will be cornerstone for guiding plane- and component-specific research activities, as well as for planning and undertaking future integration, validation and evaluation activities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

3PP	3rd Party Platform
ABE	Attribute Based Encryption
AI	Artificial Intelligence
AIDS	Anomaly Intrusion Detection System
BLE	Bluetooth Low Energy
CoT	Chain of Trust
CTI	Cyber Threat Intelligence
DB	Database
DID	Decentralized Identifiers
DLT	Distributed Ledger Technologies
eSIM	Embedded Subscriber Identity Module
eUICC	Embedded Universal Integrated Circuit Card
FE	Functional Encryption
FL	Federated Learning
GSMA	Global System for Mobile communications Association
HIDS	Host Intrusion Detection Systems
HE	Hardened Encryption
IDPS	Intrusion Detection and Prevention System
ICS	Industrial Control Systems
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IoC	Indicator of Compromise
IoT	Internet of Things
IPR	Intellectual Property Rights
IT	Information Technologies
KPI	Key Performance Indicator
OTA	Over-the-Air
OWASP	Open Web Application Security Project
PCA	Protection Control Agent
PII	Personally Identifiable Information
R&I	Research & Innovation
RATS	Remote Attestation Procedures
RIA	Resource Inventory Agent

RoT	Root of Trust
RSP	Remote SIM Provisioning
SE	Secure Element
SHDM	Self-Healing Decision Manager
SIDS	Signature Intrusion Detection System
SIM	Subscriber Identification Module
TCP	Transmission Control Protocol
VDR	Verifiable Data Registry
W3C	World Wide Web Consortium

1 INTRODUCTION

The development and increasing adoption of a plethora of Internet of Things (IoT) technologies, devices and services has had - and will continue having - a profound impact in society. Several industries (e.g., eHealth, utilities) have been disrupted and empowered thanks to IoT (e.g., enabling remote monitoring in an efficient way), while simultaneously many aspects of our everyday life have been modified (e.g., with personal assistants, wearables), generating a huge impact on businesses, consumers, and governments. Recent forecasts[1] indicate that the number of IoT devices worldwide might nearly triple from 8.74 billion in 2020 to more than 25.4 billion IoT devices in 2030. On the other hand, we witness significant growth in the report of associated vulnerabilities[2]. This reflects into proportionally higher concerns, as existing security- and privacy-related issues remain unaddressed.

To ensure that the transformation brought by IoT will benefit all citizens in a way that warrants security and privacy, Research and Innovation (R&I) needs to target the definition and development of innovative and advanced security and privacy management mechanisms and technologies that can seamlessly be integrated in a variety of contexts and applications (across different vertical market segments/applications/use cases). To address this, ARCADIAN-IoT aims at enabling a holistic framework, which establishes vertical planes to manage identity, trust and recovery in IoT systems considering its distinct entities (persons, IoT objects or devices and apps/services), while being complemented by horizontal planes enabling advanced privacy, security and encryption in a decentralized manner. Ultimately, the framework will enable a Chain of Trust (CoT) between the diverse entities participating in IoT systems, including persons, IoT objects and related apps/services.

This deliverable presents the ARCADIAN-IoT architecture, its components and associated (internal and external) interfaces. Following the adopted software system architecture specification methodology – explained in the next subsection -, the document provides further information regarding the architecture's high-level information models, and describes the preliminary intentions regarding deployment, development, operational and concurrency views.

1.1 Methodology

"Software architecture is the fundamental organization of a system embodied in its elements, relationships, and in the principles of its design and evolution[3]". The way fundamental is defined, depends on several aspects, including the target stakeholders. The main stakeholders with respect to ARCADIAN-IoT architecture include cybersecurity (spanning its different domains such as application security, data security, identity access management, network security equipment or infrastructure protection) solution providers and IoT service providers (particularly but not exclusively in digital services, the ICS, or in the medical/health domain).

The devised functional architecture is the result of an iterative process, starting from the original project concept, and the previously documented components' characteristics and associated requirements. This process was then driven by several work sessions for discussing and obtaining

[1] https://claroty.com/2h21-biannual-report/
[2] https://venturebeat.com/2022/03/03/report-34-increase-in-security-vulnerabilities-for-iot-and-it-tech/
[3] https://www.iso.org/standard/50508.html

relevant input from the consortium stakeholders (i.e. both technical partners and respective domain experts / solution providers), to agree on scientifically and operationally relevant concerns and address the needs of each domain.

ARCADIAN-IoT framework architecture introduced in this document presents not only the components of the platform but also the way they interact. The functional view of the architecture is presented in Section 2.3. To conduct the specification and description the ARCADIAN-IoT framework architecture taking into account the different stakeholders' concerns or points of view the consortium adopted the guidelines proposed by Rozanski & Woods [1]. As such, the description of the ARCADIAN-IoT framework architecture is organized into multiple viewpoints to capture different perspectives, which had to be considered during the architecture design. The several viewpoints according to this methodology include: Functional, Information or Concurrency Viewpoints, Software Structure, Deployment View, Development View and Operational View, as identified in Figure 1.

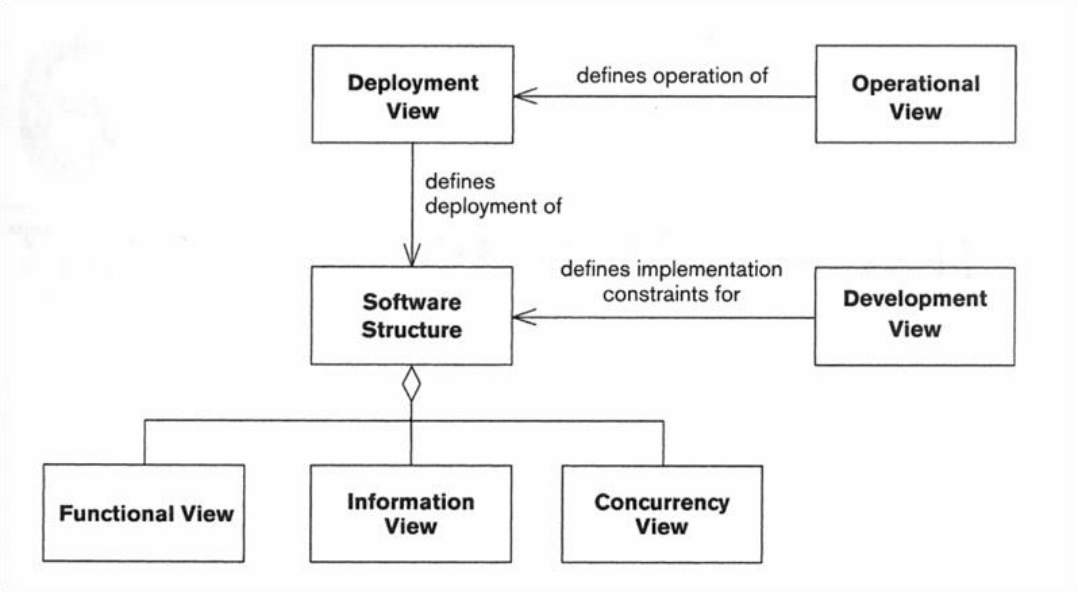

Figure 1 - Rozanski & Woods viewpoints and respective relationships [1]

All the viewpoints, with the exception of the Software Structure, are considered (at different levels) in this document, as the latter is directly dependent on stable implementation and software choices across all components. Thus, considering the R&I nature and stage of the project, this viewpoint is not to be considered.

At this stage, the functional view is the most advanced among all views, as it directly relates to the presentation of ARCADIAN-IoT architecture – the main objective of this deliverable. Nevertheless, the information gathered and decisions taken so far, allow us to provide a high-level description of the information and development viewpoints. The remaining viewpoints will be further refined as the component development and ARCADIAN-IoT Framework integration works start (within the scope of WP5). The primary objective of each viewpoint at this stage of the project can be defined as follows:

- **Functional viewpoint:** to define the system functional elements (i.e., ARCADIAN-IoT components), their responsibilities and the primary interactions with other elements –

presented in Section 2.3 (depicting the functional architecture) and Section 3 (introducing the components).

- **Information viewpoint:** to depict the way that information will be stored, distributed, and managed in ARCADIAN-IoT Framework – presented in Section 4.
- **Development viewpoint**: to describe the first implementation options and constraints registered by all partners – presented in Section 5.1.
- **Deployment viewpoint:** to portray the preliminary views of how the ARCADIAN-IoT Framework can be deployed – presented in Section 5.2.
- **Operational viewpoint:** to provide preliminary considerations about how the system can be operated, administered, and supported – presented in Section 6.1.
- **Concurrency viewpoint:** to provide a high-level overview of the component concurrency structure (e.g., synchronism or dependency) of the system, and identify components that can be executed concurrently and how this can be coordinated and controlled – presented in Section 6.2.

1.2 Document Structure

The remainder of this document is presented as follows:

Section 2 starts by introducing the ARCADIAN-IoT concept, describing what the framework encompasses and explaining what its horizontal and vertical planes consider. Furthermore, this section presents ARCADIAN-IoT Architecture in its functional view. This approach allows readers to understand from the very beginning the rationale behind the identified ARCADIAN-IoT Framework, its planes and its supporting components.

Section 3 presents ARCADIAN-IoT components, their objectives, main functionalities, and respective interfaces with other components. The description will allow the reader to better understand the role that each component has within the ARCADIAN-IoT Framework. Furthermore, a general architecture overview of each component is equally presented.

Section 4 starts by providing insight on ARCADIAN-IoT main data flows (i.e., what is exchanged between ARCADIAN-IoT components), the communication mechanisms that the framework will support, as well as a discussion on the envisioned technological choices for data storage and management. Furthermore, section 4 presents an overview of how the three ARCADIAN-IoT domains - (A) Emergency and Vigilance; (B) Grid Infrastructure Monitoring; and (C) Medical IoT – relate to ARCADIAN-IoT Architecture and its components.

Section 5 describes the preliminary development and deployment preferences that ARCADIAN-IoT consortium partners have expressed on a custom-made EU Survey, specifically devised to collect feedback from each technical partner. The aspects covered include module code management, preferred programming languages, CICD adoption, development methodology, minimum hardware requirements, packaging, among others.

Section 6 presents the known operational and concurrency aspects associated with ARCADIAN-IoT framework and its components. These aspects cover the administration, maintenance, and support processes of ARCADIAN-IoT framework.

Section 7 concludes the document with a summary of the main conclusions of the results achieved so far and provides an overview of the next activities, that include component development and overall ARCADIAN-IoT framework integration and validation.

2 ARCADIAN-IOT FRAMEWORK

This section starts by providing an overview of ARCADIAN-IoT concept, its objectives and respective KPIs. It is followed by a description of ARCADIAN-IoT vertical and horizontal planes for managing identity, privacy, trust, security and privacy. Finally, the functional architecture and deployment view are presented.

2.1 ARCADIAN-IoT Concept, Objectives and KPIs

The concept of ARCADIAN-IoT represents an integrated approach to manage identity, trust, privacy, security and recovery of IoT devices, persons and services. It relies on specialised components (presented in Section 3) laid out on vertical and horizontal planes (described in Section 2.2) to address those aspects. The vertical planes cover identity, trust and recovery management. The horizontal planes oversee managing privacy and security across the framework.

Figure 2 depicts ARCADIAN-IoT Concept, its horizontal and vertical planes, as well as the components that support the framework. The different entities (i.e., persons, IoT devices and apps/services) covered by ARCADIAN-IoT enable a straightforward way to interact with IoT systems and its operations (e.g., data collection, data processing, or data transmission) in a safe, secure and privacy-preserving manner.

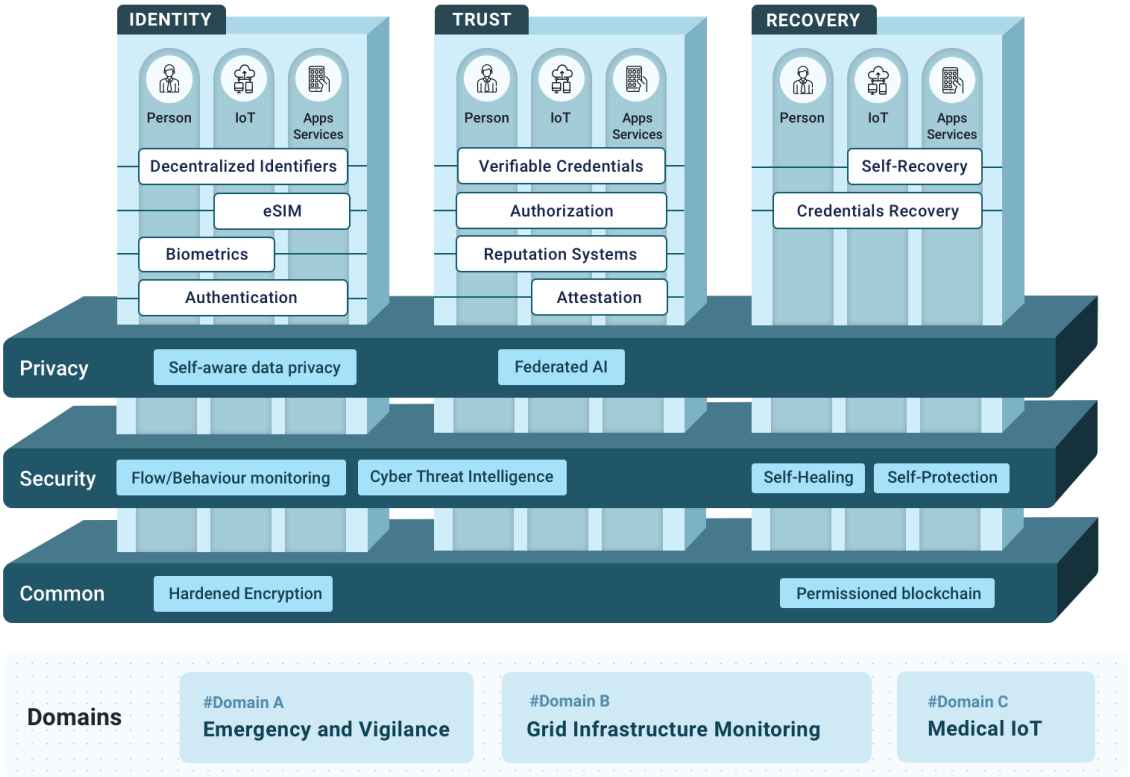

Figure 2 - ARCADIAN-IoT conceptual representation and respective validation domains.

The main objective of ARCADIAN-IoT is to enable a holistic framework with components leveraging Federated AI, Distributed Ledger Technologies (DLT), functional encryption, eSIM technologies, Cyber Threat Intelligence (CTI), and several other approaches for autonomous trust, security and privacy management for IoT systems. There are seven specific objectives, with individual KPIs. The objectives are as follows:

Objective 1: To create a decentralized framework for IoT systems - ARCADIAN-IoT framework.

- The Key Performance Indicator of this objective comprises the achievement of all the ensuing objectives and their KPIs.

Objective 2: Enable security and trust in the management of objects' identification.

- Support, at least, two identity approaches at hardware level (eSIM and CryptoChips).
- Avoid single trusted entities through decentralized approaches.

Objective 3: Enable distributed security and trust in management of persons' identification.

- High accuracy in facial recognition AI models (above 90 %) validated in real scenarios.
- Facilitate deployment of blockchain technologies by non-cybersecurity experts in Cybersecurity training sessions with, at least 20 participants.
- Interoperability with at least one eIDAS identity schema.
- Enable, at least 3 simultaneous identification approaches for persons.

Objective 4: Provide distributed and autonomous models for trust, security and privacy – enablers of a Chain of Trust.

- Enable federated AI mechanisms for, at least three heterogeneous devices and entities.
- Enhance robustness of AI models for trust and security management by a factor of 30% in real scenarios.
- Enable detection of anomalous behaviour with accuracy of 90%.
- Availability of trust evaluation models for heterogeneous entities (devices, services, persons).

Objective 5: Provide a hardened encryption with recovery ability.

- Provide at least three encryption mechanisms with low overhead.
- Enable efficient encryption with Root of Trust (RoT) information.
- Support selective recovery ability in encryption mechanisms: who and what can be recovered.

Objective 6: Self and coordinated healing with reduced human intervention.

- Recovery of at least 95% of the system functionalities prior to anomalous behaviour.

- Support coordination of recovery to pre-defined trust levels.
- Reduce human intervention to the strictly required, in healing and recovery procedures.

Objective 7: Enable proactive information sharing for trustable Cyber Threat Intelligence and IoT Security Observatory.

- Promote sharing of IoT threat data in EU, respecting privacy and data regulations.
- Enable a novel automated and privacy-preserved CTI approach exploiting the European MISP platform (MISP4IoT).

ARCADIAN-IoT will be evaluated under three different domains: (A) Emergency and Vigilance; (B) Grid Infrastructure Monitoring; and (C) Medical IoT. **Domain A** involves drones for emergency and vigilance scenarios; **Domain B** considers an early monitoring secure solution for grid (energy & utilities) main circuits of industrial and public critical infrastructures; and **Domain C** considers a secure tele-medicine solution that intends to improve the monitoring of cancer patients during active treatment process. The evaluation, planned for the project's WP5 (Use Case Implementation, Integration and Validation), will take place with the integration of ARCADIAN-IoT components in real IoT solutions.

The following section presents the ARCADIAN-IoT Planes, their interactions and respective components.

2.2 ARCADIAN-IoT Planes

ARCADIAN-IoT establishes vertical planes to manage identity, trust, and recovery in IoT systems considering its distinct entities (persons, IoT objects and apps/services) as well as horizontal planes with components that enable advanced privacy, security, encryption and decentralized technologies. The outcomes of each plane (as shown in Figure 2), together with trusted hardware providing RoT, significantly enhance the identity, security and trust of IoT systems. Moreover, ARCADIAN-IoT also sets a CoT between distinct entities of IoT systems, including persons, IoT devices and associated apps or services. CoT aims to ensure that the sources of claims (e.g., certificate) are trusted and legitimate.

The following sections describe ARCADIAN-IoT vertical (i.e., Identity, Trust, Recovery) and horizontal planes (Privacy, Security and Common functionalities), and identify the components that act within each plane.

2.2.1 Vertical Planes

The vertical planes are responsible for identity, trust and recovery management.

2.2.1.1 Identity plane

ARCADIAN-IoT enables the management of entities' identities considering their type and common mechanisms that are transversal to all identities. This plane will provide a global and interoperable identification scheme regardless of the IoT system, through **Decentralized Identifiers**, **eSIM**, **Biometrics**, and **Authentication** components. The Identity Plane should maintain its uniqueness (i.e., entities are identified in a unique way despite the system where they are used), controllability (i.e., to be able to control identity and how Personally Identifiable

Information (PII) is accessed and shared), decentralization (i.e., entities do not rely on a central issuing authority) and persistence (identity existence does not depend on the existence of an underlying company).

The Decentralized Identifiers (DID) component enables a decentralized approach for the identifiers of the diverse ARCADIAN-IoT entities. DID is a type of identifier that is globally unique, resolvable, cryptographically verifiable and does not rely in a central authority (i.e. single point of failure, target for potential attacks), and ARCADIAN-IoT will leverage it while complying with existing European identity frameworks.

The eSIM component relies on the SIM technologies, which are well accepted as secure enablers for devices identification in mobile networks. This hardware-based secure element has the necessary information and processes to securely identify subscribers in the networks to which they connect. These secure identification mechanisms will be extended to identify devices and people in IoT services.

The Biometrics component adds another factor of identification to ARCADIAN-IoT Framework. It allows to identify persons, relying on their biometrics such as face characteristics - thus supporting facial recognition.

ARCADIAN-IoT Authentication component will combine and articulate the 3 different authentication mechanisms described above - (1) DID, leveraging Self-Sovereign Identities (SSI); (2) Hardware-based Network Credentials (via eSIM); and (3) Biometrics (via facial recognition) – towards a novel robust authentication system for devices and people.

ARCADIAN-IoT authentication component will combine several authentication factors towards a novel robust authentication system for devices and people.

The **Identity plane intersects with the Privacy (horizontal) plane** regarding the privacy of information and respective authorization that is granted to data. Moreover, there is a crossover with the Security plane, to consider the security and monitoring of the involved entities and associated identities. Finally, the Identity plane equally intersects with the Common plane to leverage the hardware Root of Trust information used by the Hardened Encryption component.

2.2.1.2 Trust plane

ARCADIAN-IoT enables the trust management of entities' identities considering their type and common mechanisms that are transversal to all identities. This plane enables a global and interoperable trust management regardless of the IoT system and involved entities, being key to enable a CoT via distributed and autonomous models for trust. ARCADIAN-IoT will address trust properties such as security, privacy assurance (i.e., no unauthorized disclosure of private information), transparency (with respect to aspects such as awareness or explanation) or isolation. There are four components supporting these aspects: **Verifiable Credentials**, **Authorization**, **Reputation System** and **Remote Attestation**.

Verifiable Credentials are part of ARCADIAN-IoT's SSI solution (described in section 2.3). They will be used to ensure trust (and identity) across IoT devices, persons and services, represented by decentralized identifiers (DIDs) with associated claims.

The authorization component enables a dynamic network-based authorization approach. This component relies on updates to entities reputation scores and related security policies (from the reputation system) to automatically enforce authorization in network policy control functions (e.g., from 5G networks) preventing thus compromised IoT devices information leakage and unauthorized control.

The Reputation System provides a trust model that considers the behaviour of the devices and the interactions between entities. The Reputation System Component interacts with other ARCADIAN-IoT components to gather events' information regarding the interactions between entities in the distinct domains. Distributed Trust and Reputation Management Systems will be implemented to avoid single point of failure regarding the reputation information and will also rely on multi-level reputation scoring systems based on dominance relationships, on alpha-beta models.

The Remote Attestation component enables functional and remote attestation within ARCADIAN-IoT. Its goal is to enable peers to decide whether to consider a device is trustworthy or not. ARCADIAN-IoT aims to support hardware-based Attestation combining Trusted Platform Modules (TPMs) and Hardened Encryption, and Root-of-Trust leveraging an eSIM's secure element or cryptographic chips.

Naturally, the **Trust plane intersects with the Privacy and Security (horizontal) planes**, as trustworthiness comes hand in hand with security and privacy assurances.

2.2.1.3 Recovery plane

ARCADIAN-IoT enables the recovery management of data in case of incidents. In the ARCADIAN-IoT framework, recovery is related to a set of attributes. All critical data is secured following privacy recommendations. Recovery execution is highly autonomous, critical data is backed up (always encrypted) on servers, and data on trusted servers and devices can only be decrypted by authorized persons. This plane's components also feature properties such as efficiency, adaptability, and secure key management. There are two components in this vertical plane: the **Credentials recovery** and the **Self-Recovery** component.

The Credentials Recovery component provides a highly automated recovery of the credentials after incidents (e.g., device loss). In most cases, establishing credentials between users and backend services is a manual procedure (e.g., the user must provide his or her identity card). This procedure may be repeated in the event of an occurrence, but it renders the system ineffective. To address such issue, the Credentials Recovery component investigates how to provide an automated reestablishment mechanism for device's W3C Verifiable Credentials. In this way, devices will be able to be safely, and securely reintroduced into the system.

The Self-Recovery component aims to provide rapid recovery of data and services after incidents. The credentials recovery action must be completed first to enable the recovery. The data recovery process can only begin if the trust between the device and the backend service has been restored. Data is recovered by the self-recovery component using an encrypted backup. Aside from data recovery, the component (i.e., self-recovery) will allow for the delegation of functional encryption keys, allowing for the decryption of various layers of data. System administrators or incident response teams may, for instance, be given keys that only allow for limited decryption.

It is essential to have autonomous self-protection and self-recovery mechanisms that allow to protect or recover functionalities and data to pre-defined trust levels with reduced human intervention. Nevertheless, for that to happen, it is necessary to autonomously detect anomalous behaviour on IoT devices and related services. Therefore, the **recovery plane intersects with the security (horizontal) plane** (for attack detection information) and **with the common (horizontal) plane** via hardened encryption and distributed ledger technologies to enable trustworthy and distributed recovery mechanisms.

2.2.2 Horizontal Planes

The horizontal planes are responsible for managing privacy of data, security of systems and decentralized storage through blockchain technologies.

2.2.2.1 Privacy plane

The ARCADIAN-IoT framework enables the privacy management of confidential data and sensitive data. The goal of this plane is to provide a global and interoperable privacy management regardless of the IoT system and which is fully compliant with data privacy regulations. Two main components are focusing on providing additional privacy assurances in IoT based scenarios, the **Self-Aware Data Privacy** and the **Federated AI** component.

The Self-Aware Data Privacy component will be developed to empower the users to enhance data privacy, in particular by allowing the issuing of privacy policies for data (e.g., anonymization, pseudo-anonymization and encryption) leveraging crowdsourcing and historical information.

The Federated AI mechanisms aim to enable trust and offer automation mechanisms to further respect user privacy rights. Also, with the purpose of minimizing privacy risks associated with private data leaving trusted devices. Federated AI will equally leverage distributed Machine Learning (ML) models to enable effective cybersecurity solutions compliant with privacy regulations.

To enhance trust and privacy of the devices, services and persons, ARCADIAN-IoT privacy guarantees should be addressed across the board. Particularly, the **Privacy plane greatly intersects the identity and trust (vertical) planes**.

2.2.2.2 Security Plane

The ARCADIAN-IoT framework includes a security plane with AI-based mechanisms dedicated to attack detection in devices and networks. In IoT devices, a lightweight Host Intrusion Detection and Prevention System (HIDPS) is established by combining Federated AI-based detection and self-protection mechanisms. The network attack detection capability is complemented with the ability to mitigate the impact of attacks through self-healing and self-protection measures, forming a robust Network Intrusion Detection and Prevention System (NIDPS). This plane is set out to be composed by a total of six components: (1) **Network Flow Monitoring**, (2) **Behaviour Monitoring**, (3) **Cyber Threat Intelligence**, (4) **Network Self-Healing**, (5) **Device Self-Protection** and (6) **Network Self-Protection**.

The **Network Flow Monitoring** monitors the IoT network infrastructure for malicious flows in real-time, triggering any healing and protection actions deemed necessary. The former are performed via the **Network Self-Healing** component, which detects and mitigates network anomalies, while the latter are executed by the **Network Self-Protection**, which enforces protection rules at the data plane for safeguarding the infrastructure, IoT devices and services against volumetrics attacks.

The **Behaviour Monitoring** monitors and processes IoT devices information (e.g. system calls) for anomaly detection, while leveraging federated AI for privacy-preservation, and may trigger **Device Self-Protection** to enforce existing protection policies. Communicated threats will processed / consumed by the **Cyber Threat Intelligence** component, for generating enriched threat characterization, contributing to IoT threat landscape.

These ARCADIAN-IoT components cross over the Security Plane to the Identity Plane, which aim to oversee the devices and third-party services to trigger any security actions that are deemed

necessary to secure the devices'/persons' identification, data and operations.

The **crossover between the Security plane and the Recovery (vertical) planes** includes the device preparation for security incidents, which starts at the detection and is followed by the execution of protection mechanisms – meaning that if the device is operational, it takes actions to recover from the incident.

2.2.2.3 Common plane

The common plane provides functionalities to all other planes. The components in charge of providing such support are the **Hardened Encryption** and the **Permissioned Blockchain**.

Innovative encryption technologies are available due to the functionalities provided by the Hardened Encryption component. This component provides encryption with RoT information to ARCADIAN-IoT ecosystem leveraging secure elements (either the eSIM secure element (eUICC) or an encryption chipset / Cryptochip), and exploiting benefits from both functional encryption – allowing different subjects to retrieve different and partial information from encrypted data – and attribute-based encryption – allowing the encryption of messages for a specific received relying on its public key. The IoT devices and middleware will encrypt and decrypt data at rest using hardened encryption functionalities while the key management will be decentralized and integrated with the Permissioned Blockchain.

The Permissioned Blockchain, one of the horizontal components of the ARCADIAN-IoT system, offers decentralized data management with immutable auditability and traceability, properties leveraged for storing reputation information and for the cryptographic material used for self-recovery mechanisms.

2.3 ARCADIAN-IoT Architecture

The goal of this section is to provide a holistic view on the ARCADIAN-IoT architecture, its building blocks, components, interdependencies among components and related constraints such as adopted development tools and methodologies.

In the current stage of the project, with both baseline domain-specific use cases [2] and individual component requirements [3] established - enabling contextual guidance - performing such specification is crucial to stabilize the design of each component, e.g., how it materializes in terms of functionality and internal subcomponents, as well as associated interfaces (with other ARCADIAN-IoT components particularly but not exclusively) or APIs. This is a necessary milestone (MS2) which paves the path for the focused research and implementation (within WP3 and WP4) - where individual technical and scientific achievements will be pursued - and later on its integration in pilots (WP5) - where prototyping enables demonstration in relevant environments, Nevertheless, it must be pointed out that, while the deliverable is formally the last architecture, it isn’t a final version, as it is expectable that developments and results from the Horizontal and Vertical planes, as well as from the integration efforts (WP5), will lead to relevant updates and improvements.

As previously mentioned, the consortium adopted the guidelines proposed by Rozanski & Woods [1] to develop and describe the ARCADIAN-IoT architecture. As such, the description of the ARCADIAN-IoT framework architecture is organized into multiple viewpoints to capture different perspectives, considered during the architecture design.

For the sake of overall readability and organization, the functional and deployment views of the architecture are presented before the discussion of the viewpoints (i.e., functional, information,

concurrency, development, deployment and operation). This offers the reader with a first overview of the platform components, their interactions and their intended deployment location.

ARCADIAN-IoT is comprised of approximately 20 technical components (described in section 3) that interact with each other to offer an innovative approach to manage, in an integrated way, identity, trust, privacy, security and recovery in IoT systems supporting different types of entities, namely: persons interacting with IoT systems, IoT devices (sensors, actuators or both) and apps/services processing the collected data.

Figure 3 depicts the functional view of the ARCADIAN-IoT Framework. It shows how the technical components (represented in boxes) interact with each other (i.e., the relationships between components). The interactions, represented by directional arrows, also indicate the direction of the information flow. To ease the schema representation, at this stage, the interactions with external entities and actors are not yet represented, thus the interaction with different IoT domains is abstracted. The external entity represented in the Figure 3 depicts other CTI entities that can bidirectionally communicate with ARCADIAN-IoT's own CTI component.

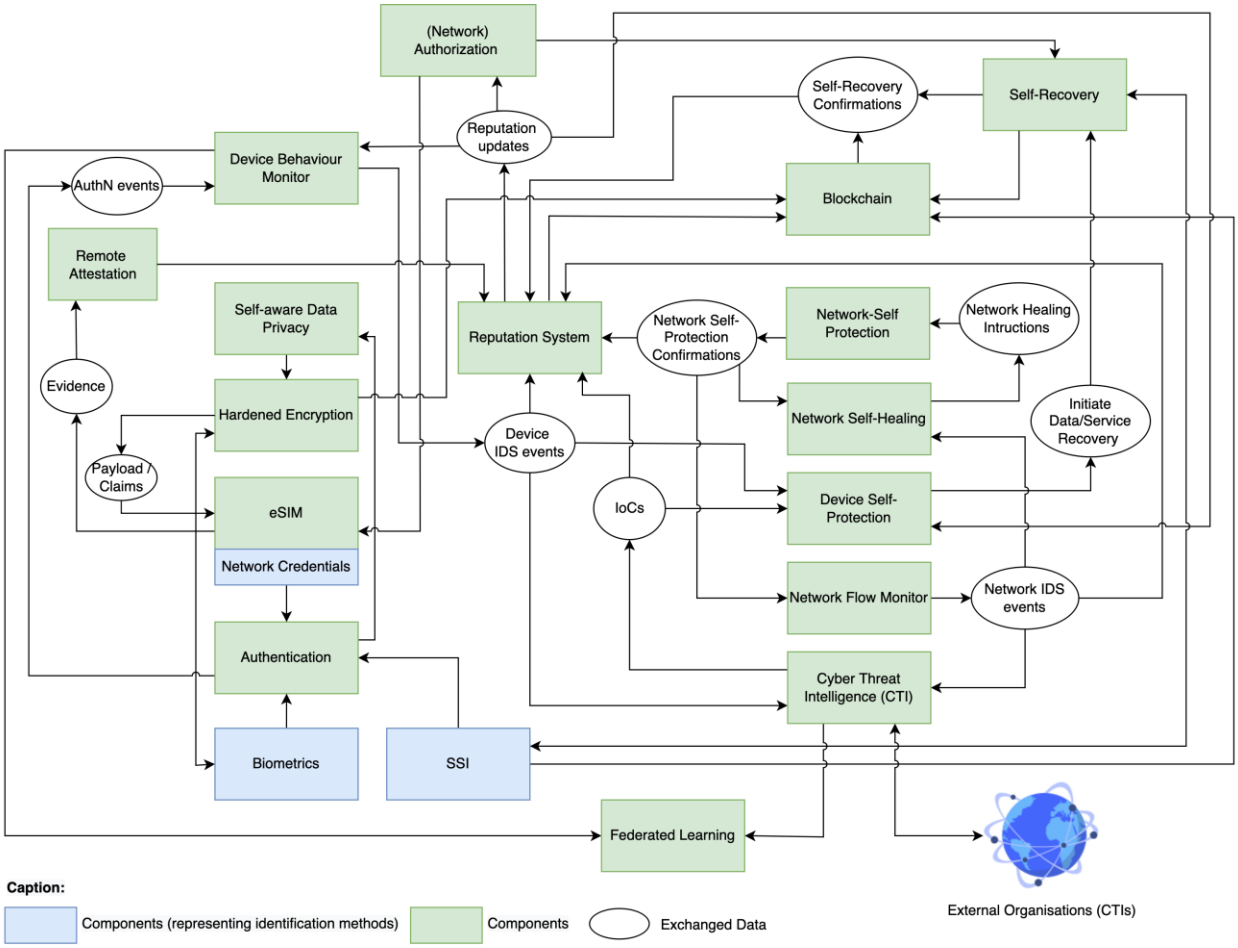

Figure 3 - ARCADIAN-IoT Architecture – Functional View

The white rounded balloons represented between each component indicate the kind of information that is shared among them. For instance, the Behaviour Monitor generates Device Intrusion Detection (IDS) events. These events are then consumed by the Reputation System, the Device Self-Protection and the CTI. The representation allows to take some conclusions regarding the flow of information and the optimal communication mechanisms to support it; for

instance, Information which is directed towards multiple boxes will in most cases be supported via a publisher/subscriber mechanism (e.g., RabbitMQ); other components establishing a one-to-one communication links will typically use other communication mechanisms such as REST APIs. The communication mechanisms are further discussed in Section 4.1.

The blue boxes represent the 3 distinct identification methods supported by ARCADIAN-IoT framework: Network Credentials, Biometrics and Self-Sovereign Identify (SSI). The latter (i.e., SSI) considers Decentralized Identifiers (DIDs) and Verifiable Credentials (VC) and aims at supporting eIDAS Bridge[4] within the European Self Sovereign Identity Framework (ESSIF) project, so that a service can issue VCs to a user.

Correlating the figure with the project's concept and objectives, some observations can be drawn. For instance, a CoT enabled by autonomous models for trust, security and privacy can be seen as resulting from the myriad of information feeding the Reputation System, both "negatively" / in case of events (e.g. device and network IDS events, failed attestation results, IoCs) and "positively" / after recovery from incidents (i.e. after device self-recovery or successful remote attestation, network self-protection confirmations), and whose computed reputation scores (consumed by the (network) Authorization, Behaviour Monitoring and Device Self-protection) lead to the enforcement of established policies across devices and network.

To complement the functional view of ARCADIAN-IoT architecture, it is also important to represent the deployment view, depicting the distribution of components and their functionality across the different deployment locations. For example, Hardened Encryption has a sub-component that is deployed directly on the IoT Device, and another sub-component (i.e., Hardened Encryption – Key Management Service) that is deployed outside the IoT Device (e.g., in the Cloud or in any other location such as the premises of the applicability domains).

ARCADIAN-IoT components can, for the sake of simplicity, be grouped in three ways: (1) On-device components, if their functionality is assumed to be fully provided in the device; (2) Multi-deployment components, in case their functionality (i.e., subcomponents) is somehow split or its deployment spans both the device and a remote location (i.e. the network or cloud); and (3) Cloud/Network-based components, if their functionality is fully ran outside of the device. Figure 4 presents the high-level deployment view and the respective deployment location that is envisioned for each component.

The functional and deployment views grant the reader the essential knowledge of ARCADIAN-IoT components, their interfaces, the main data exchange and the high-level deployment locations. This is complemented by an introduction of each component (and the explanations of their interfaces) in the next section. Additionally, further considerations about the deployment locations will be provided in Section 5 where more details about the deployment options and respective locations will be presented. It is worth mentioning that the presented architecture is not final, as some further specifications will be required to encompass some specific needs such as the integration of existing IoT systems (a need expressed e.g. within domain B). For such cases, ARCADIAN-IoT is currently considering the adoption of a Proxy Adapter or Gateway (e.g. for enabling performing custom functionality such as re-encryption or proxying of data or requests between IoT devices and 3rd party end users, as required in Domain B). Any relevant update to the architecture will be latter presented in the scope of WP5.

[4] https://joinup.ec.europa.eu/collection/ssi-eidas-bridge

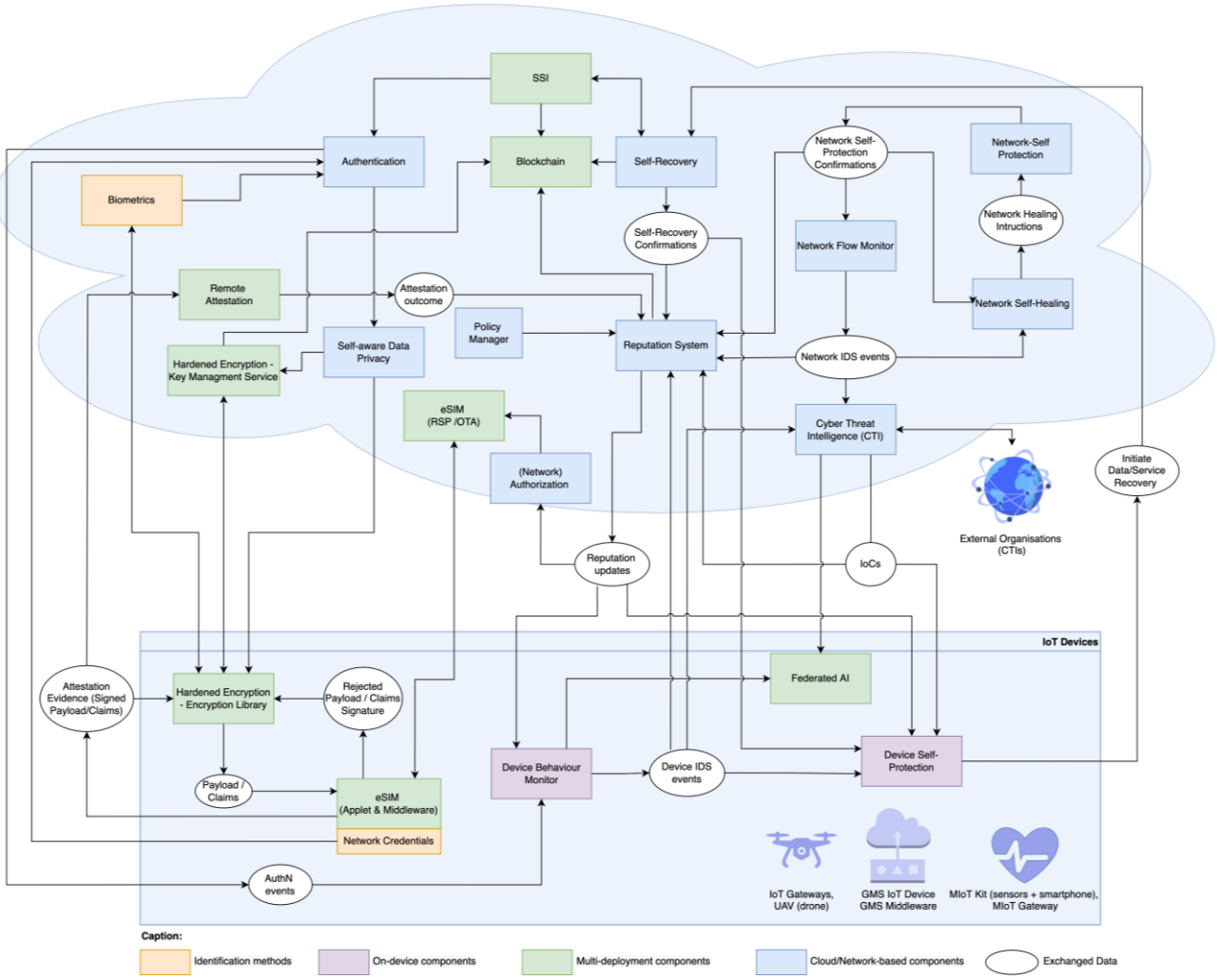

Figure 4 - ARCADIAN-IoT Architecture – High-level Deployment View

The following section will describe in more detail the role and interfaces of each of ARCADIAN-IoT component.

3 ARCADIAN-IOT COMPONENTS (PER PLANE)

This section provides an overview of all ARCADIAN-IoT components as per the framework's planes. The goal of this description is to present the main objectives of each component, as well as to identify their internal and external interfaces thus, further detailing the overall architecture with the functionalities provided by each component.

The components are grouped by their placement within the six ARCADIAN-IoT Planes:

- **Identity**: Decentralized Identifiers, eSIM, Biometrics, Authentication;
- **Trust**: Verifiable Credentials, Authorization, Reputation System, Remote Attestation;
- **Recovery**: Self-Recovery, Credentials Recovery;
- **Privacy**: Self-Aware Data Privacy, Federated AI;
- **Security:** Network Flow Monitoring, Behaviour Monitoring, Cyber Threat Intelligence, Self-Healing, Network Self-Protection and Device Self-Protection;
- **Common**: Permissioned Blockchain, Hardened Encryption and Cryptochip.

3.1 Identity plane

The identity plane is composed of 4 components, presented in the following sub-sections:

- Decentralized Identifiers (led by ATOS)
- eSIM (led by Truphone (TRU))
- Biometrics (led by UWS)
- Authentication (led by TRU)

3.1.1 Decentralized Identifiers

Decentralised Identifiers (DID) are described in the World Wide Web Consortium (W3C) DID Core Specification [4] as "… a new type of identifier that enables verifiable, decentralized digital identity. A DID refers to any subject (e.g., a person, organization, thing, data model, abstract entity, etc.) as determined by the controller of the DID. In contrast to typical, federated identifiers, DIDs have been designed so that they may be decoupled from centralized registries, identity providers, and certificate authorities. Specifically, while other parties might be used to help enable the discovery of information related to a DID, the design enables the controller of a DID to prove control over it without requiring permission from any other party. DIDs are URIs that associate a DID subject with a DID document allowing trustable interactions associated with that subject.".

Each DID document expresses cryptographic material, verification methods, or services, which provide a set of mechanisms enabling a DID controller to prove control over the DID. Proving control over the DID enables services to provide trusted interactions associated with the DID subject.

The DID is a URI composed of three parts; scheme identifier, a DID method and a specific identifier within the DID method, and resolve to DID Documents. ARCADIAN-IoT DID solution will follow the basic architecture model as described in the DID Core specification:

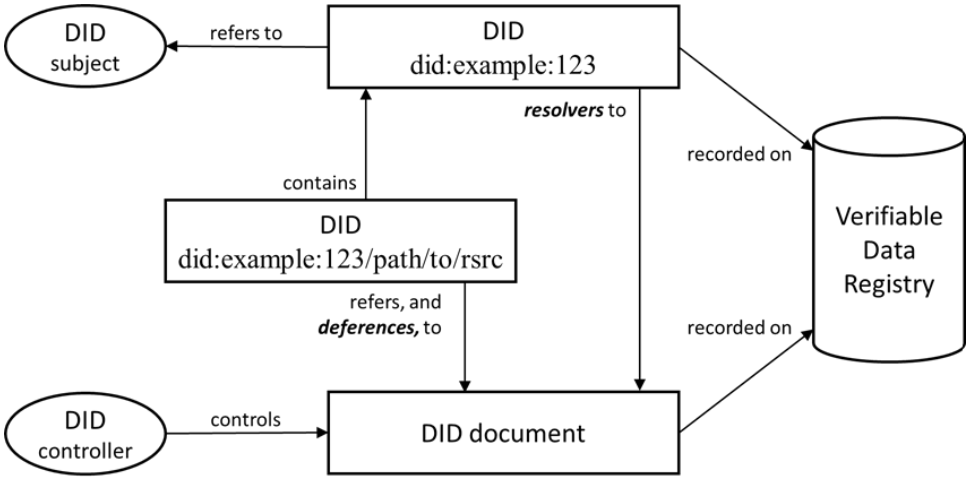

Figure 5 - ARCADIAN-IoT DID solution's basic architecture model[5]

A DID method is an implementation of the features described in the DID specifications, to answer specific needs usually recorded on a Verifiable Data Registry (VDR). It specifies the operations by which DIDs and DID documents are created, updated, recovered, deactivated and resolved. In the context of ARCADIAN-IoT the DID Documents will be stored on a VDR. Candidate VDRs under analysis include distributed ledger technology and also sidetree-based DID Method overlay networks composed of independent peer nodes. These nodes implement Content Addressable Storage to host the DID Docs and interact with a decentralized system to provide a notarised trust anchor, as described in the Sidetree specification [5]. In this approach the components are deployed outside of the ARCADIAN-IoT service and follow a decentralized architecture.

Internal interfaces

- Sidetree trust anchor: If a sidetree implementation is used an internal interface based on a client application invocation of a smart contract is needed to notarise the DID Operations on a Sidetree overlay network.

External interfaces

- DID Operations:
  - DIDs support a variety of DID operations, which require the DID owner to first generate specific data values and cryptographic material, which can be done locally on the device. The different types of operations on the DID Doc, are listed below:
    - Create (to publish the DID Doc);
    - Update (to update the DID Doc);
    - Recover (to recover control of the DID);
    - Deactivate (to deactivate the DID).

    Note: If these operations will finally be implemented by the Sidetree network then the specification [5] provides a standards-based support for them, otherwise they will be specific to the VDR technology.
- DID Resolution:

[5] https://www.w3.org/TR/did-core/

- Additionally, the implementation should support the resolution of the DID in the ecosystem to resolve to a DID Doc, as per the DID Resolution specification [6].

- DID Communication:
  - For agents to be able to provide applications with a secure, private communication methodology built on top of decentralized design of DIDs[6]. This enables agents to reliably exchange DIDs and verify each other as the holder of that DID and to reliably share Verifiable Credentials, all with cryptographic proofs based on the DIDs.

    Note: DIDCOMM provides a standard based protocol that is needed to be supported by devices and services alike so that they can successfully interwork with each other [7].

3.1.2 eSIM

The eSIM expands the already well-accepted and widespread SIM card technology to an embedded format with remote provisioning and management capabilities. With the purpose of fostering security by design, the eSIM will act, in the context of the ARCADIAN-IoT, simultaneously as:

- Secure **connectivity enabler** for devices and people;
- **Secure Element** (SE) capable of storing identity and authentication credentials at devices hardware level and use them in ARCADIAN-IoT multi-factor authentication process (section 3.1.4);
- **RoT** with the ability to contribute to hardened encryption and attestation processes with evidence that allow to infer trust;
- **Local (edge) authorization agent** (section 3.2.2) with specific actions of self-protection and self-recovery according to devices threat level / trustworthiness.

Having as hypothesis the use and extension of state-of-the-art GSMA eSIM specifications for using the eSIM as RoT, such as IoT SAFE, the component will “establish end-to-end, chip-to-cloud security for IoT products and services”[7], while acting as a “the hardware Root of Trust in an IoT device”[7]. When present in personal devices (devices proven to belong to a person, e.g., a smartphone), the eSIM (the credentials stored in it) will be used to securely identify and connect persons to services or to other persons.

Existing connectivity applets and IoT SAFE guidelines will be applied as starting point in this component research and development. The result is expected to be a novel SIM applet (and related middleware), the ARCADIAN-IoT eSIM applet, which should be able to, not only provide connectivity, but also contribute to novel processes of identification and authentication of devices and persons in Cloud services, and to mechanisms of hardened encryption, attestation, devices’ self-protection and devices’ self-recovery.

[6] https://www.w3.org/TR/did-core/, Last accessed 2022-04-28

[7] IoT SAFE Executive Summary (https://www.gsma.com/iot/wp-content/uploads/2020/05/IoT-SAFE-Executive-Summary.pdf); , Last accessed 2022-04-28

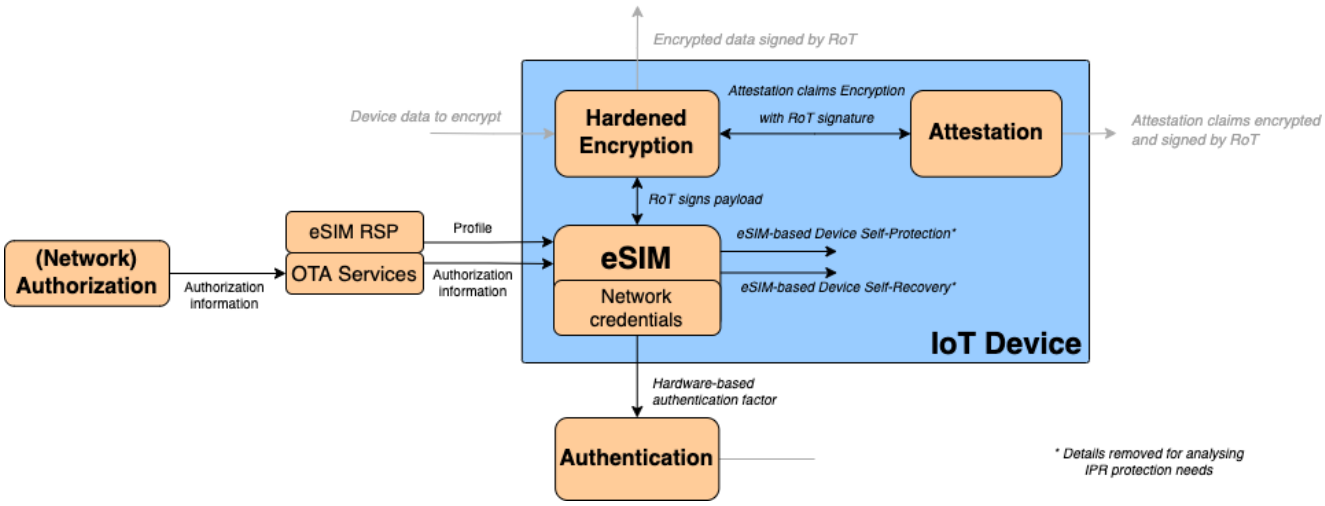

Figure 6 - eSIM overall architecture view

Figure 6 depicts the eSIM interfaces with several other ARCADIAN-IoT components, showing thus most of eSIM-related research objectives. These are explained below in the external interfaces. The interfaces that are internal, from the eSIM ecosystem (RSP and OTA), despite not being target of research in the project context are presented due to being enablers of the research to be done.

Internal interfaces (within the eSIM ecosystem, according to GSMA Security Accreditation Scheme[8]):

- Profile provisioning: Remote SIM Provisioning (RSP) to the devices eUICC for the ARCADIAN-IoT eSIM profile (with the mentioned security applet) download and enablement process;
- Over-the-Air (OTA) communication services: secure communication services with the devices' secure element.

External interfaces (with other ARCADIAN-IoT components):

- Hardened Encryption (section 3.6.1): Interface for requesting the eSIM to sign an encrypted payload or a communication session. The request is expected to be made by the Hardened Encryption component present in the device firmware or app.
- Authentication (section 3.1.4): Interface for using network credentials (stored in the hardware secure element - eSIM/eUICC) for authentication in third-party services. The interface is expected to be with a specific service present in the core network, which is able of confirming the network credentials of a given communication and forward the result to the ARCADIAN-IoT Authentication component.
- Remote Attestation (section 3.2.4): Interface for requesting the eSIM to sign attestation claims (trustful evidence that the payload was sent by that device). The request is expected to be made by an attestation subcomponent present in the device firmware or app; alternatively, as depicted in the figure above, attestation claims will be encrypted before being signed, thus being perceived as encrypted payload from the eSIM point of

[8] GSMA Security Accreditation Scheme (SAS) (https://www.gsma.com/security/security-accreditation-scheme/); Last accessed 2022-03-16

view (using the Hardened Encryption interface).

- Authorization (section 3.2.2): OTA eSIM interface for receiving, in the ARCADIAN-IoT applet, device trustworthiness information from Authorization component. The information is used for eSIM-based device self-protection and eSIM-based device self-recovery.

3.1.3 Biometrics

The biometrics component adds another factor to identify persons, relying on their biometrics such as face characteristics. This component will support facial recognition considering operational challenges. The Biometrics component will be responsible for receiving a set of photos from the client with the face that constitutes the database (DB) to perform face identification. The biometric data will be in a secured database where only the biometrics component will have access. Besides, the component will process the video feed received by the drone. Internally, the component will run a face identification algorithm producing an output if the faces available in the scene are identified against those available in the DB. In addition, we will explore the possibility to provide together with identification result, the bounding box that represents the part of the scene where the face has been identified to allow a GUI to create an overlay figure. Figure 7 shows the main interfaces foreseen for this component.

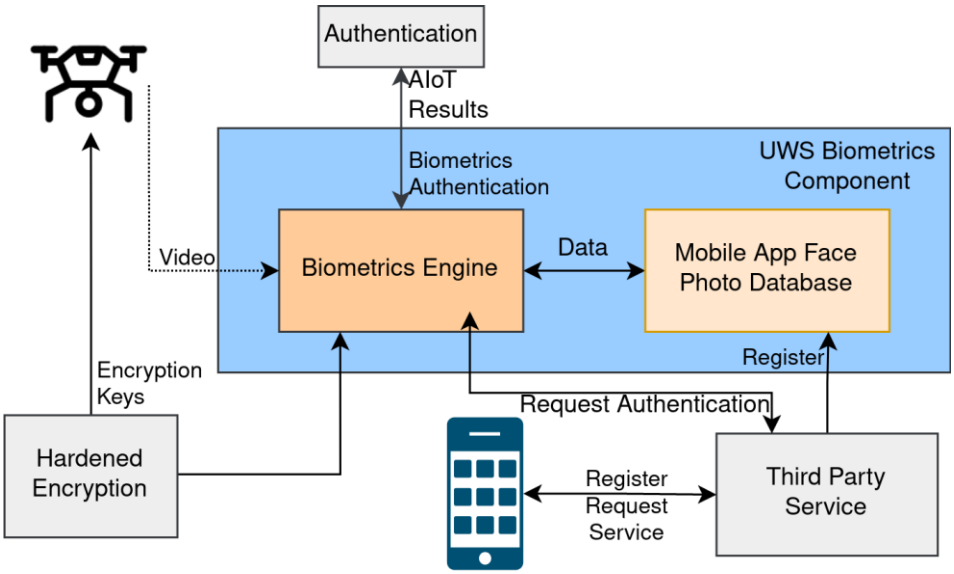

Figure 7 - Biometrics Architecture Model

Internal Interfaces:

As shown in Figure 7, the Biometrics Engine interfaces with the Mobile App Face Photo Database. The image can be stored in two ways in the Mobile App Face Photo Database:

- The complete Image: The Mobile App Face Photo Database could not be persistent to be protected against privacy violations and thus will be a temporal memory structure that will be discarded as soon as it is no longer needed. The third-party service should feed the database each time and an ID is needed.
- The output of the Face ID: The Mobile App Face Photo Database could just store the features of the Face ID algorithm and not the image itself.

External interfaces:

- Drone Video Feed will be an H264 encoded video that will be delivered using RTP or RTMP protocol from the drone;

- Third Party Service:
  - It will be an interface to send the photo encoded as a JPEG/PNG/RAW or similar and an ID of the person to the third-party service. This service will share the photo and the ID to the Mobile App Face Photo Database for the client registration;
  - Third party service will request authentication sharing the client ID and the Biometrics engine will execute the identification algorithm over the video streamed.
- Authentication: An interface to send the results of the face detection algorithms using a JSON format and provide credential verification and validation.

3.1.4 Authentication

ARCADIAN-IoT authentication component will combine several authentication factors towards a novel robust authentication system for devices and people. The authentication factors, to be used simultaneously whenever possible, are: (a) Network credentials stored in the device's hardware (secure element – eSIM/eUICC – section 3.1.2) that will be used to identify and authenticate ARCADIAN-IoT entities (persons or devices) in third-party services; (b) Decentralized self-sovereign approaches whose credentials used for identification and authentication are not stored in any centralized element (section 3.1.1); (c) novel biometrics mechanisms (section 3.1.3). The simultaneous use of 2 (for devices) or 3 (for people) of these factors will provide a robust authentication framework for the project. ARCADIAN-IoT authentication component will aim to manage these factors, contributing, as result, for the access control to ARCADIAN-IoT compliant third parties services and data.

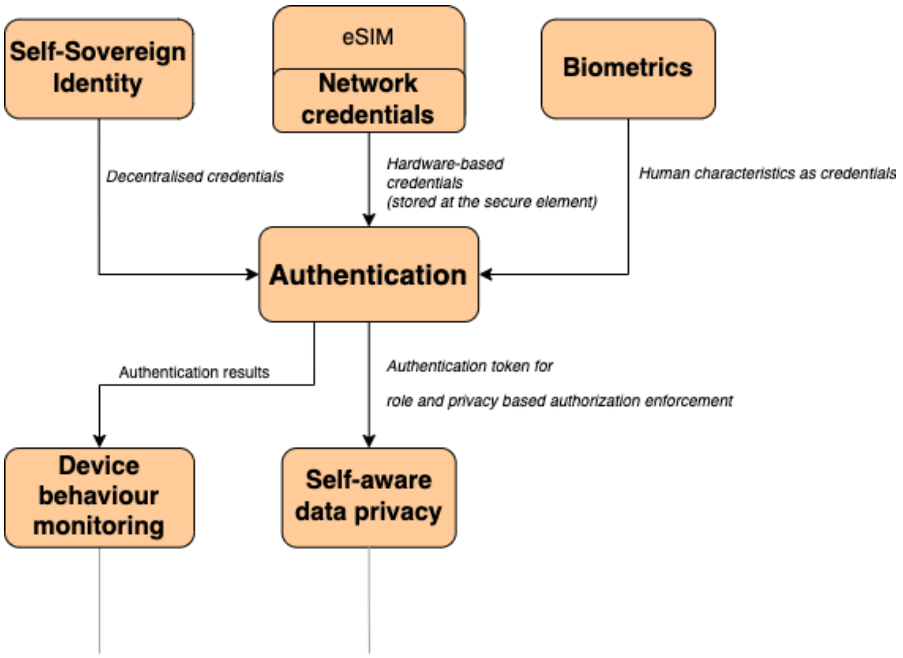

Figure 8 - Authentication architecture view

The figure above shows that from the authentication will result a token for role-based and privacy-based authorization enforcement at the Self-aware data privacy component (section 3.4.1), and information for behaviour monitoring (section 3.5.2).

Considering the existent identification and authentication factors, for the moment, it is assumed that services will just be identified within the framework with a decentralized approach.

External interfaces:

- Decentralized identifiers (self-sovereign identity) (section 3.1.1): secure decentralized credentials provisioning and verification.
- Network credentials (section 3.1.2): secure hardware-based credentials provisioning and verification.
- Biometrics (section 3.1.3): secure credentials provisioning and verification.
- Self-aware data privacy (section 3.4.1): token with the authentication result to enforce role-based and privacy-based authorization.
- Device behaviour monitoring (section 3.5.2): authentication result for behaviour monitoring purposes.

3.2 Trust plane

The trust plane is composed of 5 components, presented in the following sub-sections:

- Verifiable Credentials (led by ATOS)
- Authorization (led by TRU)
- Reputation System (led by IPN[9])
- Remote Attestation (led by IPN)

3.2.1 Verifiable Credentials

ARCADIAN-IoT will provide a Self-Sovereign Identity management solution that is built on W3C Verifiable Credentials specification[10] that is a core standard that is helping to facilitate the Self-Sovereign Identity (SSI) approach in decentralizing identity management. The solution enables trusted registration of users through the issuing of identity claims as Verifiable Credentials to their respective secure crypto based digital identity wallets without depending on centralized Identity Providers with its inherent privacy risks. In turn, these credentials can be later presented by the user to services and apps which require to authenticate the user in a trusted crypto based manner that only the holder of these credentials can do (based on the fact they were issued to the user´s wallets relying on their Decentralized Identifier).

Atos have an SSI prototype asset called LedgerUself built on top of Hyperledger Aries which will be leveraged to support authentication (depicted in Figure 9). Its cloud agent is aimed to support one of the ARCADIAN-IoT authentication means for persons and devices and the Mobile App will support end users to authenticate themselves to 3rd party services (aka Relying Parties) making use of the ARCADIAN-IoT framework. Integration is further shown with the Verifiable Data Registry as previously introduced in section 3.1.1.

[9] Supported by University of Coimbra, as linked 3rd party
[10] https://w3c-ccg.github.io/vp-request-spec/

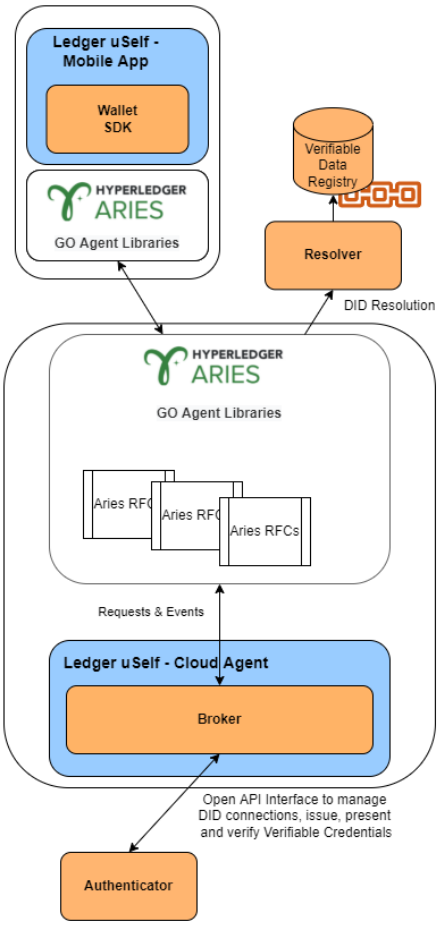

Figure 9. Ledger uSelf built on top of Hyperledger Aries GO Agent

External interfaces

- SSI Agent to SSI Agent interface to Issue, Request and Present Verifiable Credentials;
- Get VC schema;
- Verify VC is not revoked.

The SSI identity management solution will provide the core building blocks for supporting these external interfaces for issuing, presenting and verifying credentials as per the W3C model [8] shown in Figure 10.

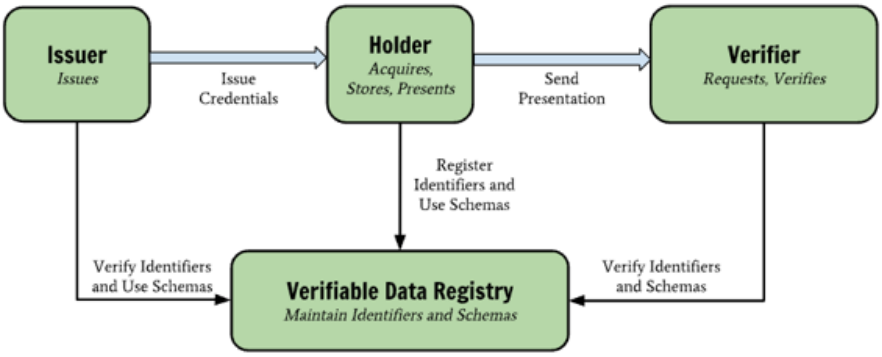

Figure 10 - W3C Verifiable Credentials Model

Internal interfaces

- Access to the DID(s) and corresponding private key(s);
- Service interface to Issue, Request and Present VCs to SSI Agent.

3.2.2 Authorization

According to the research made and the current understanding, ARCADIAN-IoT authorization factors are divided in two: (1) the self-aware data privacy (section 3.4.1), directly connected with the authentication component (section 3.1.4), which focus on user-defined privacy-related authorization policies, coupled with role-based authorization policies; and (2) a network-based authorization enforcement, focused on this section.

The (Network) Authorization component aims to develop a novel dynamic network-based authorization enforcement that will be fed with information of the trustworthiness scores from the entities present in the IoT network, and with ARCADIAN-IoT security policies that should be applied according to devices and persons trustworthiness (Reputation System – section 3.2.3). The purpose is to prevent the leakage of private/confidential data from (compromised) IoT devices to the Internet and avoid the control of IoT devices from internet services that are not found trustworthy. This action is only possible due to the technology positioning within the network core, where subscribers are identified and policy control functions (e.g., PCRF/PCEF) are applied before the traffic being routed to/from the internet. The component purpose is, therefore, to apply reputation-based policies, in a policy control system, to enforce IoT data communication permissions to/from IoT devices or persons.

Additionally, this component will also have a function to securely route the trustworthiness information of the device or person, through GSMA-SAS accredited OTA services, to the eSIM secure element. This information will allow eSIM to act, in processes of self-protection and self-recovery, as an independent secure agent inside devices.

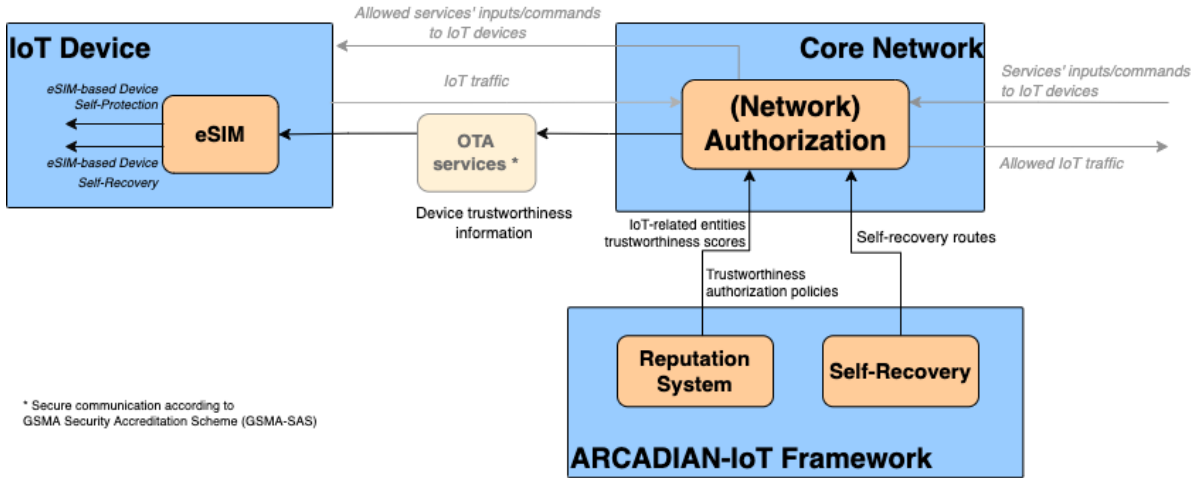

Figure 11 - Network-based authorization architecture view

Internal interfaces:

- Interface for a novel reputation-based policy generation function, which creates rules recognizable by the (Network) Authorization component based on reputation information, and sends them to the network policy control component;
- Interface for updating the set of routes allowed for devices / persons according to their reputation and self-recovery procedures;

- Interface to a network subscriber identity repository, to translate the ARCADIAN-IoT identifier to a network identifier (e.g., IMSI).

External interfaces:

- Reputation System (section 3.2.3): Interface for inbound data from the reputation system – entities reputation scores and authorization policies.
- eSIM (section 3.1.2): Interface for informing secure elements about devices trustworthiness, for actions of self-protection and self-recovery.
- Self-Recovery (section 3.3.1): Interface for receiving internet routes for recovery to be enforced for compromised devices' communications.

3.2.3 Reputation Systems

The Reputation System is a component that is responsible to devise the reputation of other components and respective entities included in the ARCADIAN-IoT architecture. The reputation score is built based on algorithms that consider dominance relationships, for instance, the overall feedback provided by users for a specific entity, like the work of [9]. In case, of such information is not available, algorithms relying on alpha-beta algorithms will be employed, as these identify events in two types: positive – which increment reputation, and negative – which decrement reputation [10]. An example of the events that can be classified as positive and negative is illustrated in the Table 1 for the DGA service in domain A.

Table 1 - Examples of events affecting reputation

Use Case as per D2.2 0/ Description	**Entity**	**Reputation Ratings Logic Initially, reputation is NULL for all entities**	**Person Reputation**	**Service Reputation**	**Device Reputation**
A1– Person registers in the DGA service and install mobile APP,	Person (identified by SSI) / DGA service	Users also installs app (download) and registers himself in the DGA Service providing all the fields in the form	+	+	+
A3- Person retrieves personal data	Person / DGA service / Device with DGA App	The DGA app on the phone is an old version, and does not support robust encryption mechanisms.			-- device and mobile app

As depicted in Figure 12, the Reputation System component has diverse interfaces and distinct internal layers.

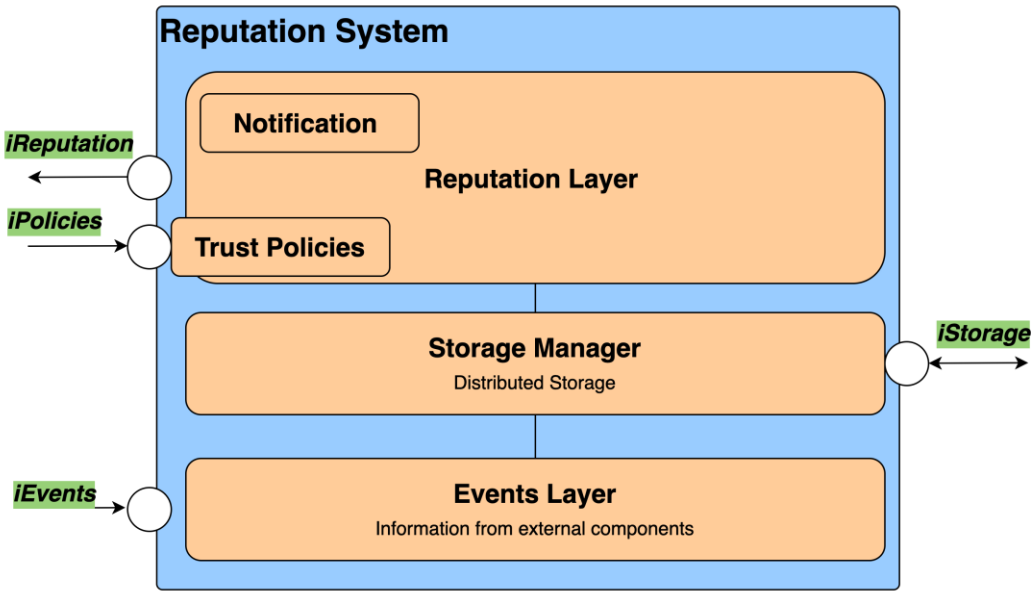

Figure 12 - Reputation System Architecture Model

Internal interfaces:

- Interface of the event layers with sub-components like the event collector and the event analyser;
- Interface of the storage manager to allow a distributed and secure storage of the reputation score;
- Interface with the reputation layer which requires information from the events layer to determine the reputation the score and communicates with the storage manager sub-component for secure storage of reputation score information.

External interfaces:

- Interface with Permissioned Blockchain to store the reputation information;
- Secure Interface to receive information from other components through a message bus (e.g., RabbitMQ). Through this interface, Self-Recovery and Network Self-Protection Confirmations are received, as well as Device IDS and Network IDS events, as well as IoCs or notifications about device Attestation Results and outcomes;
- Interface to notify other components regarding reputation modification and the trust policies that apply to the respective reputation score. This interface provides reputation updates to (Network) Authorization, Device Self-Protection and Behaviour Monitoring components;
- Interface to receive information of policies associated with a specific service in a domain, the policies must map the reputation score to specific actions, for instance to deny (network) authorization of user if reputation score is below a certain threshold.

3.2.4 Remote Attestation

ARCADIAN-IoT will support attestation of devices (e.g., IoT device or gateway, smartphone) and services. The attestation of devices intends to remotely check the integrity of devices and thus assess their trustworthiness, e.g., for authorization to access protected or sensitive information or services. The main objective is to support remote and functional attestation mechanisms

leveraging Root of Trust mechanisms, following the attestation procedures architecture[11] specified by the IETF Remote Attestation procedures (RATS) working group and enabling multiple verifiers.

In RATS, one peer (the Attester) produces believable information about itself - Evidence - to enable a remote peer (the Relying Party) to decide whether to consider the Attester a trustworthy peer or not. RATS are facilitated by an additional party, the Verifier, which appraises Evidence via appraisal policies and creates the Attestation Results to support Relying Parties in their decision process. In order to appraise Evidence generated by an Attester, the Verifier needs to trust the Attester's RoT, which, in ARCADIAN-IoT, is provided by one of the secure elements (e.g., eSIM or cryptochip). Evidence is generated in the Attester by Attesting Environments based on Claims (i.e., values and other information) collected from Target Environments (as depicted in Figure 13). The means to collect such data are diverse and may include e.g., reading system registers and variables, calling into subsystems, taking measurements on code, memory, or other security related assets of the Target Environment. Attesting Environments are responsible for formatting the Claims appropriately, and use key material and cryptographic functions (e.g., signing or cipher algorithms) to generate Evidence.

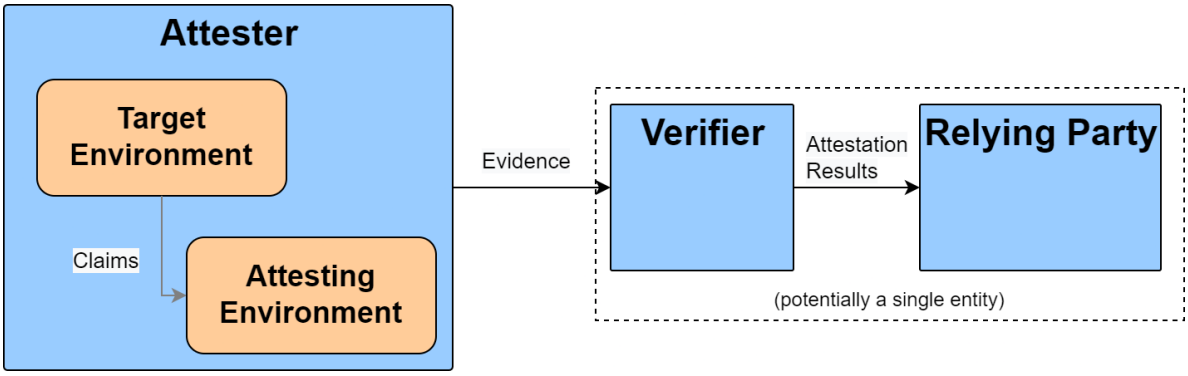

Figure 13 - Remote Attestation Architecture Model

In order to implement remote attestation procedures in ARCADIAN-IoT, the Attester entity, located in the IoT device, will 1) leverage TPMs for ensuring system (claims) integrity, 2) be supported by hardware - either eSIM (sections 3.1.2) or crypto chip (section 3.6.1.2) - for providing RoT, and 3) involve the Hardened Encryption library (section 3.6.1) to ensure confidentiality of the attestation evidence. Namely, claims (e.g., hardware model, timestamp, token ID) will be collected from the device's TPM, and then, as part of Attestation Environment functionality[13], these will be encrypted (via the Hardened Encryption library), before the associated payload being sent to the eSIM for signing and generating a signed hash. The Verifier(s) will then decrypt the received Attestation Evidence (enabled by access to the secret keys provisioned by Hardened Encryption component), validate the RoT signature, ensuring the integrity of the sender and appraise the evidence. Finally, the Attestation results will critically contribute to the Chain of Trust by feeding the Reputation System with crucial information for assessing IoT devices trustworthiness. Figure 14 depicts the current high-level assumptions on the remote attestation procedure in the project:

[11] https://datatracker.ietf.org/doc/html/draft-ietf-rats-architecture-14

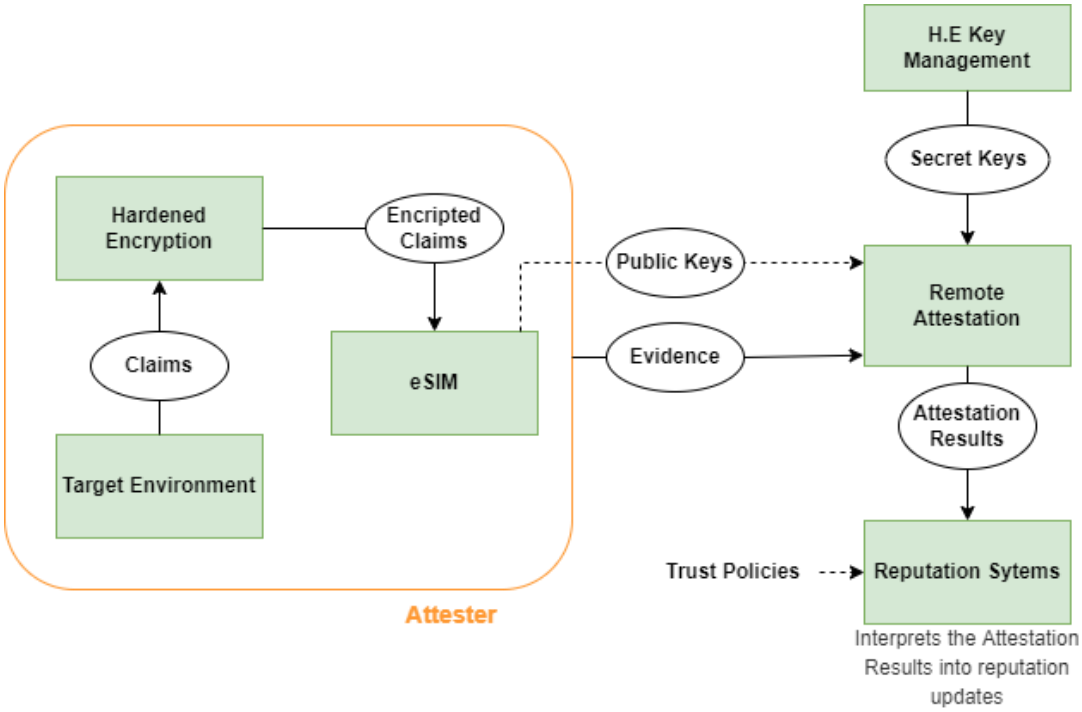

Figure 14 - Hardware-based RoT (with eSIM) Remote Attestation in ARCADIAN-IoT

Internal interfaces: the main internal interfaces of the Attestation System are:

- Attester – Verifier OR Attester - Relying Party (depending on the adopted model, i.e., Passport model vs Background Check model): used by the Attester to send evidence;
- Attester (Target Environment) - Attester (Attestation Environment): used to collect and send Claims from the device. As mentioned, Attestation Environment builds on the TPM (e.g., the eSIM) along with the necessary functionality for feeding it with the claims;
- (optional) Verifier - Relying Party OR Relying Party - Verifier: used by Verifier to send Attestation Results to Relying Party.

External interfaces:

- Reputation System (section 3.2.3): interface for sending device attestation results or a simplified form of the attestation outcome (e.g., success / failure), which are processed towards updating device reputation;
- Verifier – Hardened Encryption (section 3.6.1): interface for collecting secret keys required for functional decryption;
- Verifier – eSIM (section 3.1.2): interface for obtaining the public key.

3.3 Recovery plane

The recovery plane is composed of 2 components, described below:

- Self-Recovery (led by XLAB)
- Credentials Recovery (led by ATOS)

3.3.1 Self-recovery

The Self-recovery component comprises client-side (on-device) scripts, that allow devices to back up any relevant data, and a storage server that hosts the backups. The ability of a device to store

or recover its backups is contingent on the reputation and trust score of a device. A compromised device will not have the authorization to store or recover backups until its reputation is restored via the process of Credentials recovery. The recovery process can be initiated manually, for example by a system administrator when performing maintenance or repairs on a device, or automatically, where the Self-recovery service is triggered by an issued event from the Device Self-protection component, when an anomalous event is detected.

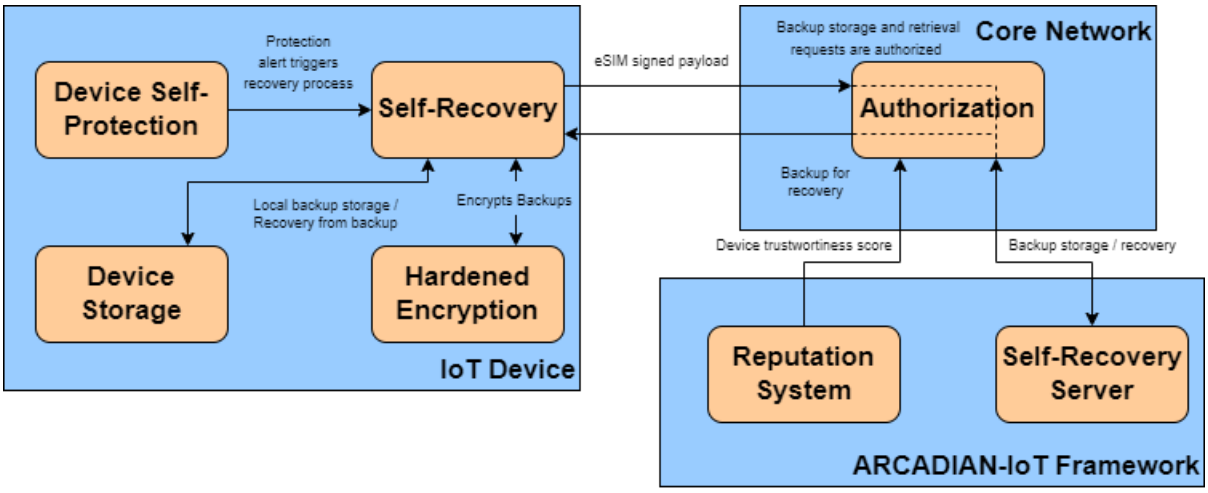

Figure 15 - Self-recovery diagram

Data backups will be encrypted before storage using the results of Hardened encryption task, preferably on the device itself, or in the case of resource constraints, an encryption proxy (depicted in

Figure 16). In case of intermittent connectivity or disconnects, devices will be able to store the backups locally before transferring them to the storage server.

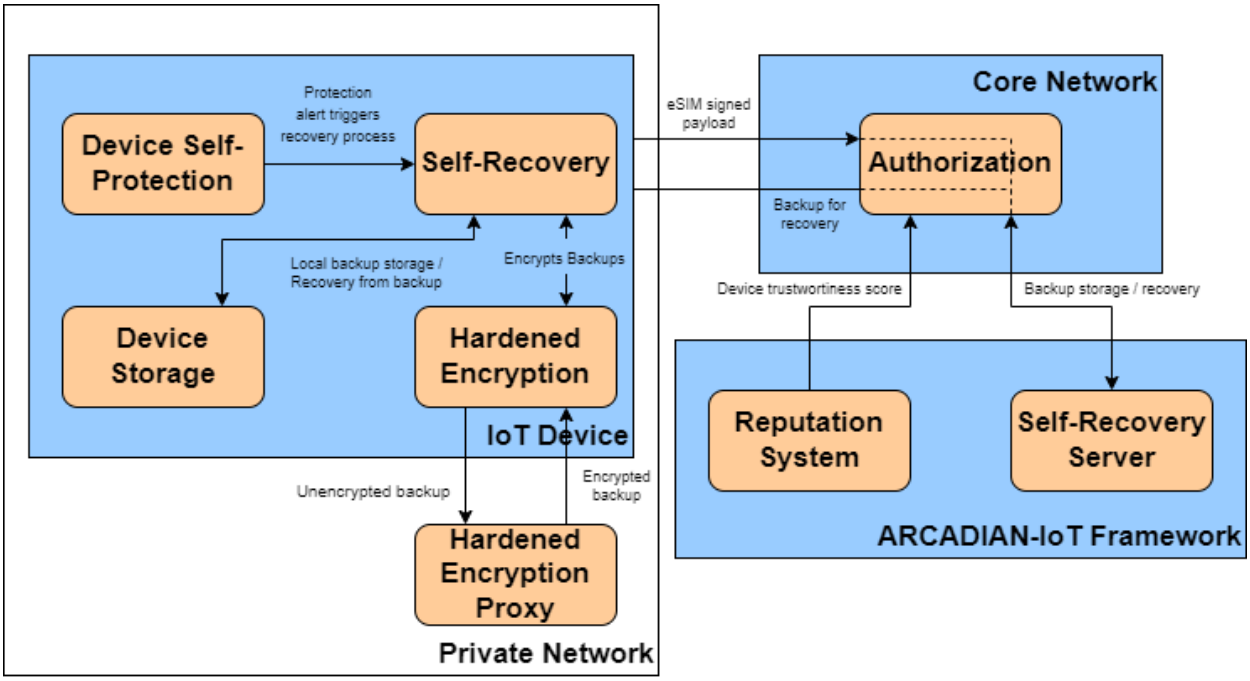

Figure 16 - Self-recovery with encryption proxy

To address concerns regarding storage of sensitive data or, more generally, data privacy, the concept of attachable storage will be investigated, where the actual backups are stored on-premises, while the ARCADIAN-IoT platform only stores backup metadata (shown in

Figure 17).

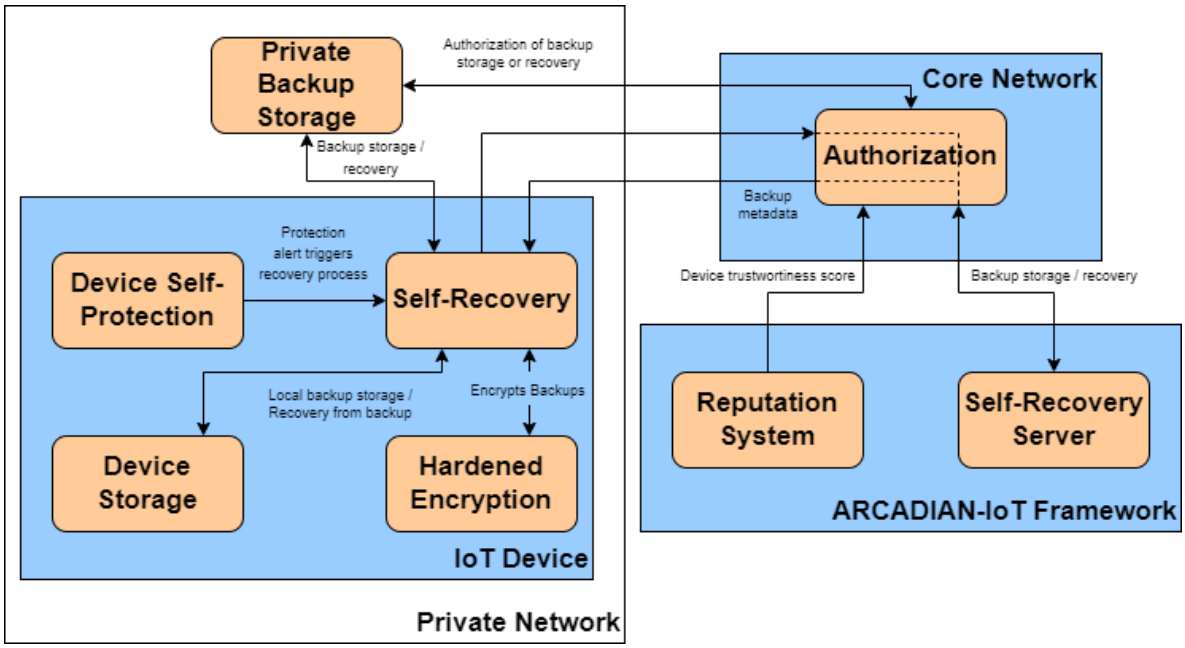

Figure 17 - Self-recovery with private storage

Besides the data self-recovery actions described, the eSIM (section 3.1.2) will consider devices / people reputation changes (recovery of a trustworthy status) to contribute to the automatic operational recovery of the device.

Internal interfaces

- Device storage for local data backups.

External interfaces

- Authorization backed up by the Reputation system to verify if a device is allowed to store or retrieve backups.
- The device self-protection component (upon the enforcement of more restrictive policies) will also notify the self-recovery component so that recovery actions can be initiated.

3.3.2 Credentials recovery

The recovery of credentials is the first and necessary step to trigger a data recovery mechanism. The secure recovery of credentials is vital as there the trust between the device and the backend services is established.

To recover Verifiable Credentials and metadata associated to wallet connections, it is needed to first perform a backup and therefore any solution should employ an encrypted backup option to an external self-recovery backup server with access provided by ARCADIAN-IoT authentication components. The figures below show the different access approaches that are able to be employed depending if it is a mobile or IoT device.

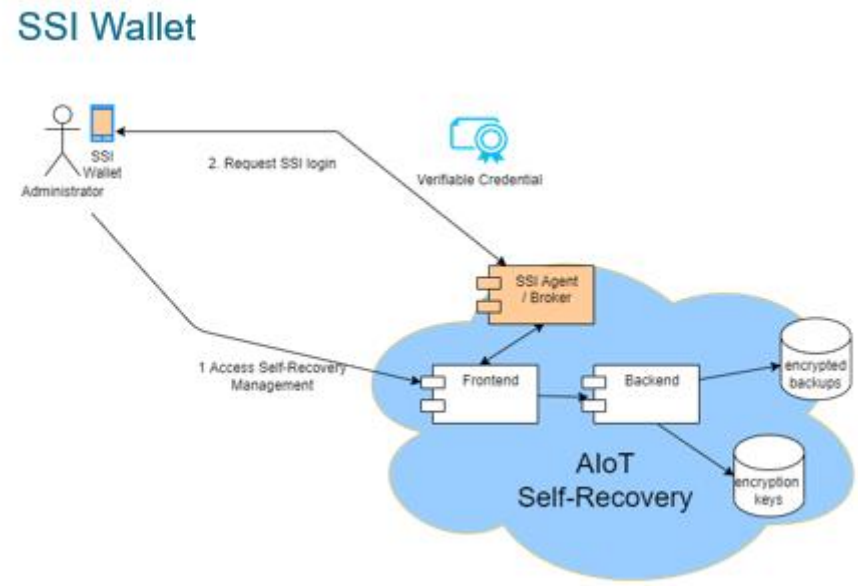

Figure 18 - ARCADIAN-IoT Verifiable Credential access to Self-Recovery

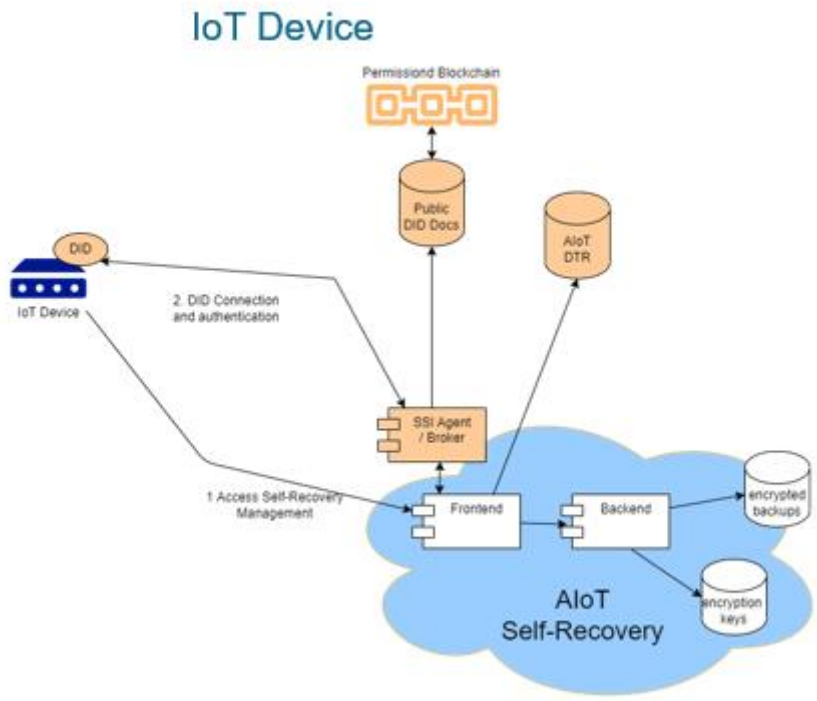

Figure 19 - ARCADIAN-IoT DID Access to Self-Recovery

Different credential recovery options are explored below depending on the device:

Self-Recovery Account Credentials Recovery for a SSI Wallet on a mobile device

- it can be setup a quorum of trusted entities trusted emails to be issued with different portions of a recovery key to be used to request a restoration of the SSI Wallet including its VCs.
- a user can be issued with a QR Code containing a recovery key to be used to request a restoration of the SSI Wallet including its Verifiable Credentials.

Self-Recovery Account Credentials Recovery for an SSI Agent on IoT Device

- To support an automated process for restoration of a device it is needed for the device to connect to the Self-recovery component with their public DID and perform DID authentication where the DID is also checked to be in the trusted ARCADIAN-IoT DTR registry. Using a public DID anchored on the ARCADIAN-IoT permissioned blockchain ensures that the device is the holder of the private key associated to the DID.
- In the case of IoT devices that lost their DID or was compromised the DID DOC would have to be first updated / recovered with the IoT device updating its associated private key.

The self-recovery backup server would need to support a client self-recovery library for the SSI Wallet/ SSI Agent to be able to encrypt / decrypt the backups it handles with the self-recovery back-up server.

Internal interfaces

- SSI Wallet/Agent updates its internal database to import or export the credential and

connection metadata.

External interfaces

- Out-Of-Band trust interface request to the Self-Recovery system for example with email sent or quorum token for being invited to be issued with a new account VC;
- SSI Agent Self-recovery system issues a backup account VC to the device once trust is re-established;
- SSI Wallet/Agent presents account VC to access the self-Recovery system and access backups for devices registered to that user;
- SSI Wallet/Agent calls the self-recovery client backup library to encrypt or decrypt the Verifiable Credentials and wallet connection metadata;
- SSI Wallet/Agent calls the self-recovery back-up server to upload or download an encrypted back-up with access control performed either on existing device DID connection or otherwise from a link sent to the user´s email address.

3.4 Privacy plane

The privacy plane is composed of 2 components:

- Self-Aware Data Privacy (led by Martel (MAR))
- Federated AI (led by RISE)

3.4.1 Self-aware data privacy

The Self-Aware Data Privacy component goals are twofold: on one hand the component aims at enforcing data privacy (e.g., anonymization or encryption) via flexible (user defined) privacy policies and on the other hand it aims at suggesting (recommending) privacy policies based on the nature of data or wisdom of crowd. Figure 20 depicts the architectural view of the Self-Aware Data Privacy component.

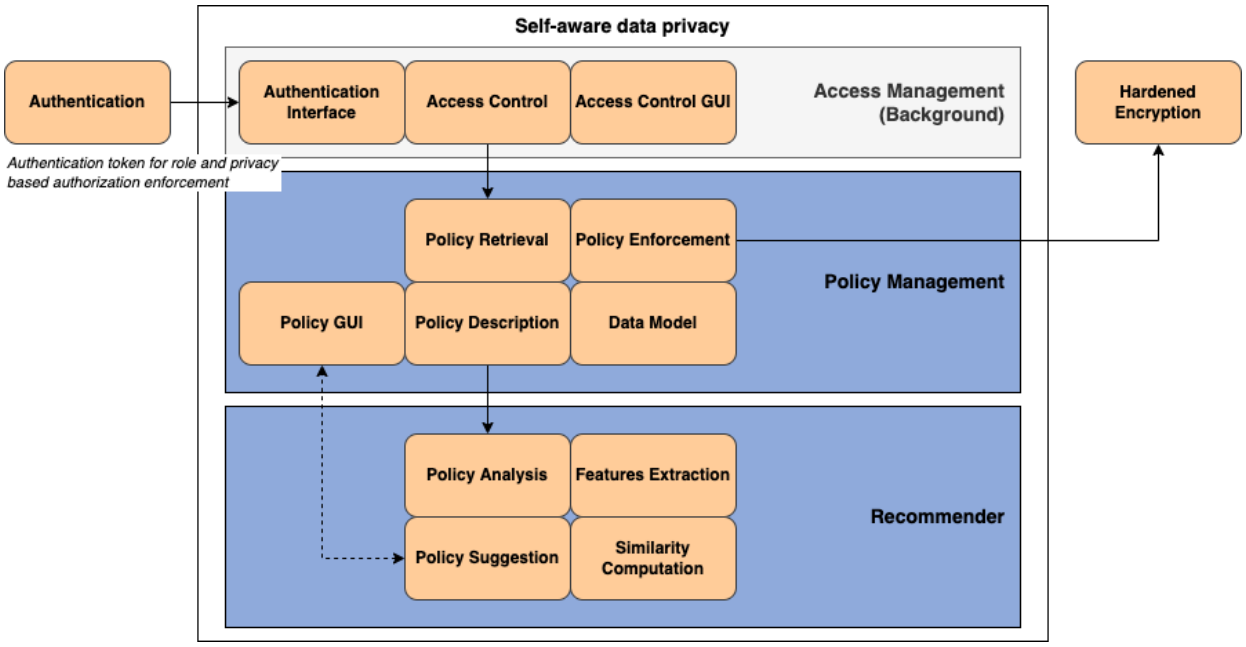

Figure 20 - Self-aware data privacy high-level view

The enforcement of the privacy will be flexible enough so that it will be applicable to different levels of data granularity, from a single attribute to a specific data type or to a particular data set.

The resource owner will always be in control of the policy definition and their maintenance, and the recommendation system will support the owner with the policy definition.

Internal interfaces

The Self-Aware data privacy component relies on Keycloak[12] (or similar OIDC providers) to manage user authentication and access to user data i.e., role and permissions and their description (Access Control). This is a background and will include a GUI. The Policy Management component will leverage OPA to process and define policies (Policy Retrieval and Description), supported by an appropriate data model and GUI for policy specifications and updates (insertion, deletion and update by the authorised users), all of which will enable the recommender system. The Policy Enforcement will allow to turn into practice the specified policies with the help of the Hardened Encryption libraries e.g. by implementing anonymisation algorithms or leveraging Attribute Based Encryption (ABE) ones.

The Recommender system will analyse the policy specification and extract the features which will allow the comparison with the previous ones and will provide suggestions to the user via the Policy GUI.

External interfaces

The Self-Aware data privacy component will leverage the Hardened Encryption libraries (section 3.4.1) to enforce the privacy policies (e.g. ABE algorithms) and protect the data and will interact with the Authentication component (section 3.1.4) to enable only authorized users to specify policies on data they are owning/controlling/processing. The interface with the Authentication component will be mediated by Keycloak (or similar OIDC providers).

3.4.2 Federated AI

This component aims to provide dependable and privacy preserving federated learning capabilities to ML-based components. Integrity of the model will be ensured both from the point of view of source integrity (no malicious participant) and data integrity (privacy of raw data and local model updates is preserved). The local model in federated AI framework will be tested as an anomaly detector in the context threat monitoring for IoT networks.

The component further provides accurate and trustworthy framework by addressing challenges of federated learning, such as imbalanced and non-IID data, and adversarial attacks. Federated AI includes two sub-components: data rebalancer and model resizing and sharing (Figure 21). Data rebalancer provides a way to rebalance and fit non-IID data to the framework; Model resizing and sharing provides a communication-efficient and robust framework for model aggregation. This sub-component accelerates and protects the local model from being attacked by adversarial attacks from malicious entities.

[12] https://www.keycloak.org/

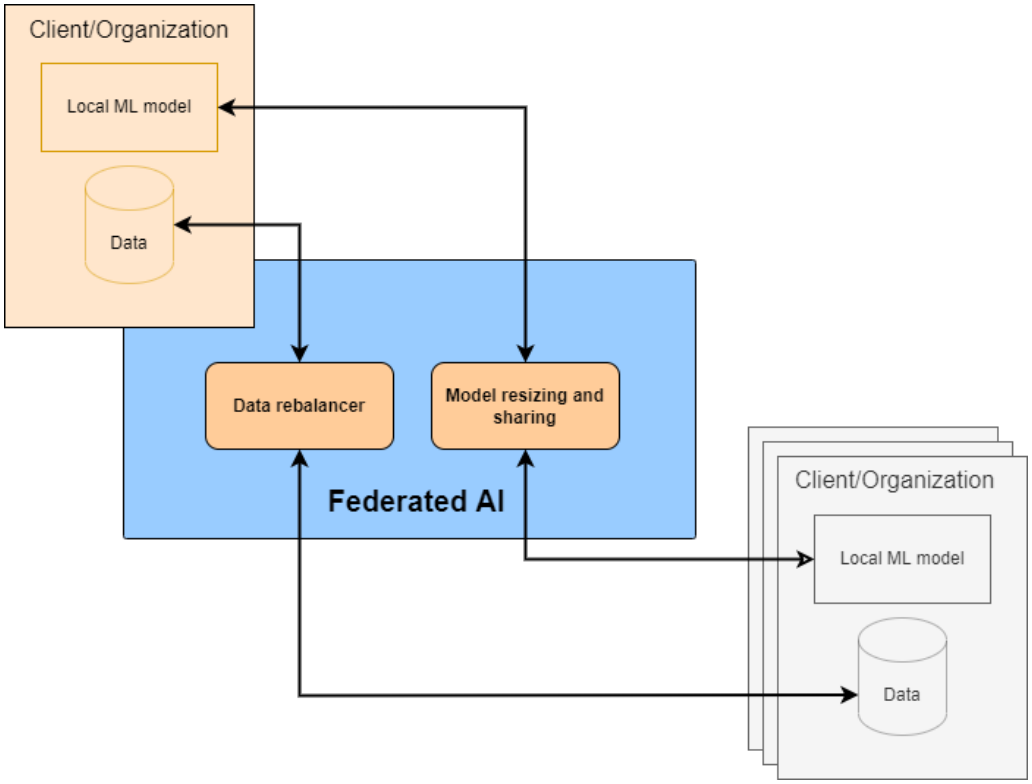

Figure 21 - Federated AI Architecture Model

External interfaces

The FL component will be developed as integrated module within those ARCADIAN-IoT components that provide AI models as their functionalities (e.g., CTI and Behaviour Monitoring component). For this reason, the "data rebalancer" sub-component will be directly interfaced with the local database while the "model resizing and sharing" sub-component will be directly interfaced with the local ML algorithm, both within the client component. At the same time, both sub-components need to communicate with other remote entities in the same or different organization.

3.5 Security plane

The security plane is composed of 6 components described below:

- Network Flow Monitoring (led by UWS)
- Behaviour Monitoring (led by IPN)
- Cyber Threat Intelligence (led by RISE)
- Network Self-Healing (led by UWS)
- Network Self-Protection (led by UWS)
- Device Self-Protection (led by IPN)

3.5.1 Network Flow Monitoring

This component is in charge of performing the detection of malicious attacks by relying on the usage of existing Network Intrusion Detection System (NIDS) such as Snort, Suricata or Bro. The NIDS will be a hook in the data plane and will apply detection rules to inform about alerts related to malicious attacks. The Security Flow Monitoring Agent (SFMA) is a new component working

as a wrapper for the NIDS to provide the following innovations:

- It will allow the application of the detection rules in overlay networks and IoT networks such as LTE-M, NB-IoT, LoRa or similar;
- It will allow the provision of fine-grain metadata about the malicious traffic with specific information about the overlay networks;
- It will also allow to provide an abstraction layer to plug in multiple NIDS technologies (e.g., Bro, Suricata, Snort).

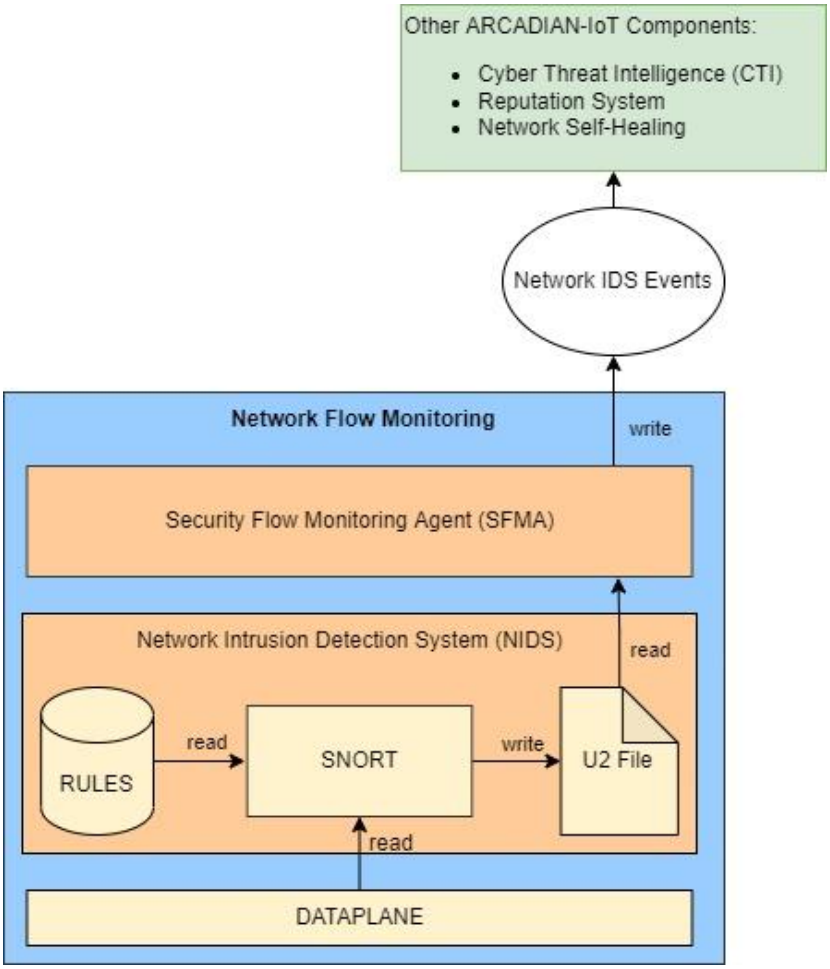

Figure 22 – Network Flow Monitoring Architecture Model

Internal Interfaces:

- There is a southbound API for the NIDS implemented as a direct hook into the data plane using PCAP, PF_RING or any similar low-level API to get access to raw data packets;
- There is an east-west API for the NIDS with a file that represents the detection rules loaded into the NDIS;
- There is a northbound API for the NIDS where the malicious alerts are reported. This API is relying on file-based sharing mechanisms using the standard security information exchange format UNIFIED2;
- There is a southbound API for the SFMA that is continuously monitoring alerts received in UNIFIED2 format.

External interfaces:

- Alert Interface will provide the results of the detection of the attack. It will use AMQP 0.9.1

protocol to send the results of the detection using a JSON format to a publication/subscription middleware such as RabbitMQ or similar. Currently considered consumers of the alerts include the Network Self-healing, Reputation System and the CTI.

3.5.2 Behaviour Monitoring

Behaviour Monitoring component is a host-based intrusion detection system (HIDS). This component takes the task of examining incoming events that are specific to the host device (such as what applications are being used, what files are being accessed, permission changes, sequences of system calls and authentication attempts). It will be comprised by a set of subcomponents that aim to collect and classify (through lightweight machine learning models) that information, in real time, to make sure that everything is running normally and detect any sign of intrusion. The resulting output, which will consist of a value that indicates if the event was classified as an intrusion or normal and a level of confidence, will be forwarded to other ARCADIAN-IoT components.

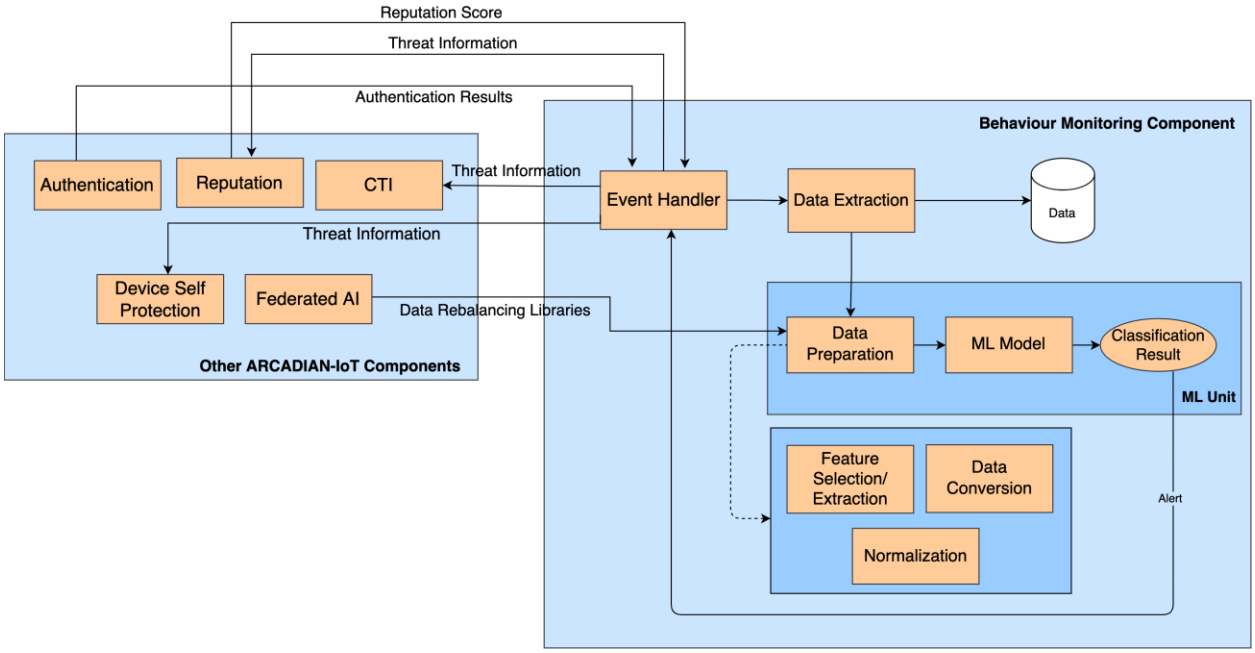

Figure 23 - Behaviour Monitoring Architecture Model

Internal Interfaces:

- Data Extraction: Extracts raw data directly from the device pertinent to user activity (e.g., system call traces)
- Data Preparation Module: The extracted raw data from the device and data from other ARCADIAN-IoT components (described in the external interfaces) is processed by this module and then sent to the detection ML model, which will output a classification of either normal or intrusion from the input data.
- Detection Module: The data from the Data Preparation Module is forwarded to the detection module, which will consist of ML models that will indicate, given an input, if it is an intrusion or not. If an intrusion is detected from the event traces, then the model will classify what type of intrusion occurred.
- Event Handler/Alarm: The resulting classification of the intrusion detection module will be packaged along with other information, such as the level of confidence and the possible

type of intrusion that occurred.

External Interfaces:

- Authentication: Interface used to receive authentication results from the Authentication component (e.g., consecutive authentication attempts).
- Federated AI: Interface used to access Federated AI libraries and model aggregation and update service.
- Reputation System: Interface designed to receive reputation updates (acting as a weighted input for the intrusion detection models) from the Reputation System and to send relevant alerts about device's behaviour to that component.
- Alerts: Interface that will provide the results of the attack detection. It will use a yet to be defined AMQP protocol to send the alerts to a publication/subscription middleware such as RabbitMQ or similar. Other components such as the CTI, Device Self-Protection and Reputation system can then subscribe to such alerts and act accordingly.

3.5.3 Cyber Threat Intelligence

The CTI component will provide an instrument to gather, produce, elaborate, and share information about cyber threats and attacks in IoT domain and affected IoT devices. Threat information will be collected and processed in terms of Indicator of Compromise (IoC) customized for IoT domain. The CTI component will be structured in four main sub-components: the open-source threat intelligence platform MISP, an IoT IDS event parsing and formatting module, the IoC aggregation sub-component, and the machine learning (ML) model manager.

The core of ARCADIAN-IoT's CTI will be the European MISP platform which will bring its simple and efficient approach for sharing threat information to the IoT domain. MISP core will be extended with a novel IoT IDS event parsing and formatting module which will analyse and process IDS events originating from IoT devices and network and enable the creation and sharing of IoC for the IoT domain. The IoC aggregation sub-component will aggregate and pre-process information coming from multiple and different IoC sources (e.g., vulnerability databases, OSINT sources, etc.). Finally, the ML model manager will manage the ML models based on threat data in the CTI and integrate the functionalities provided by the Federated AI component into the CTI which will enable privacy preservation while sharing threat information or trained models with third parties. The ML models will be used for threat data classification so as to identify additional info about the threats (e.g., level/class of compromission).

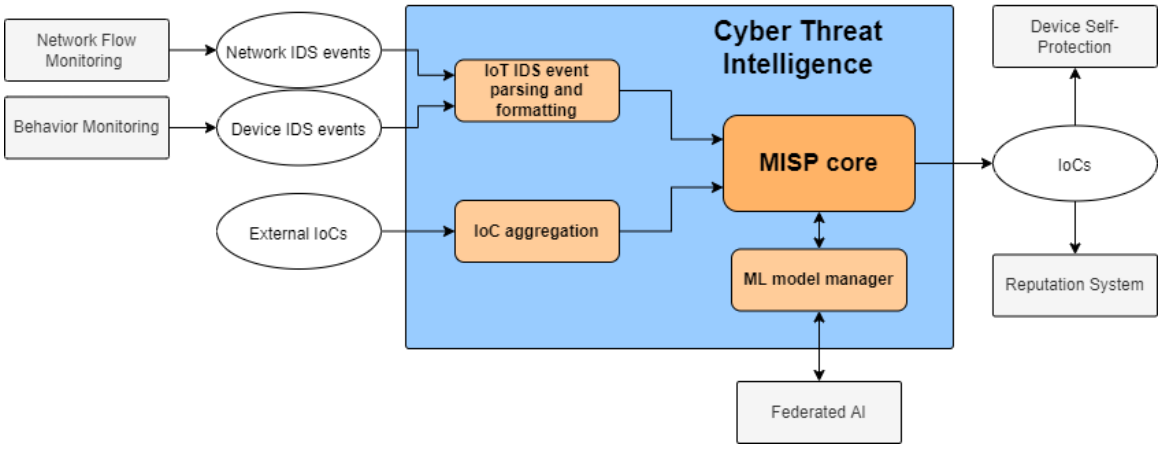

Figure 24 - Cyber Threat Intelligence Architecture Model

Internal interfaces

MISP is interfaced to the other three subcomponents:

- An interface with IoC aggregation sub-component for receiving and gathering threat data from other IoC producers and external CTIs;
- An interface with IoT IDS event parsing and formatting module processing different types of IDS events resulting from IoT devices and generated IoCs;
- An interface with the ML model manager for sharing threat information in the form of trained models.

External interfaces

- An inbound interface receiving IDS events and threat data from other components in ARCADIAN-IoT;
- An inbound interface receiving IoCs originating from other sources and external CTIs;
- An outbound interface sharing IoT-specific IoCs;
- An interface with the Federated AI component for sharing machine learning models with other CTIs.

3.5.4 Network Self-healing

The Network Self-Healing is a complex system, comprising several components and subcomponents, some of them exploiting a large number of Linux system tools. Its main purpose is to provide self-healing capabilities to the ARCADIAN-IoT infrastructure against cyber-attacks without any human intervention. To achieve so, this component relies on two components, the Network Flow Monitoring component (Section 3.5.1) in charge of the detection of malicious attacks and the Network Self-Protection (Section 3.5.5) component in charge of enforcing protection rules into the data plane. When an attack has been detected, it needs to be complemented with the topological information of the whole infrastructure to allow the self-healing component to take autonomous healing decisions taking into consideration the topology of the network. To allow so, UWS's Resource Inventory Agent (RIA) will make use of fundamental protocols and Linux inventory tools to perform the discovery of devices, interfaces and interconnections between devices and interfaces to allow an understanding of the network topology. Thus, UWS's Self-Healing Decision Manager (SHDM) will use both the topological information provided by UWS's RIA and the alerts related to malicious attacks provided by Network Flow Monitoring and will match that information against the Healing Decision Rules to autonomously determine what set of actions needs to be implemented. The set of Healing Decision Rules is predefined in the system according to the security policies defined by network administrators (policy-based approach). These actions will be sent as an intent to UWS's Network Self-Protection component (Section 3.5.5) which will enforce the self-healing of the network without any human intervention.

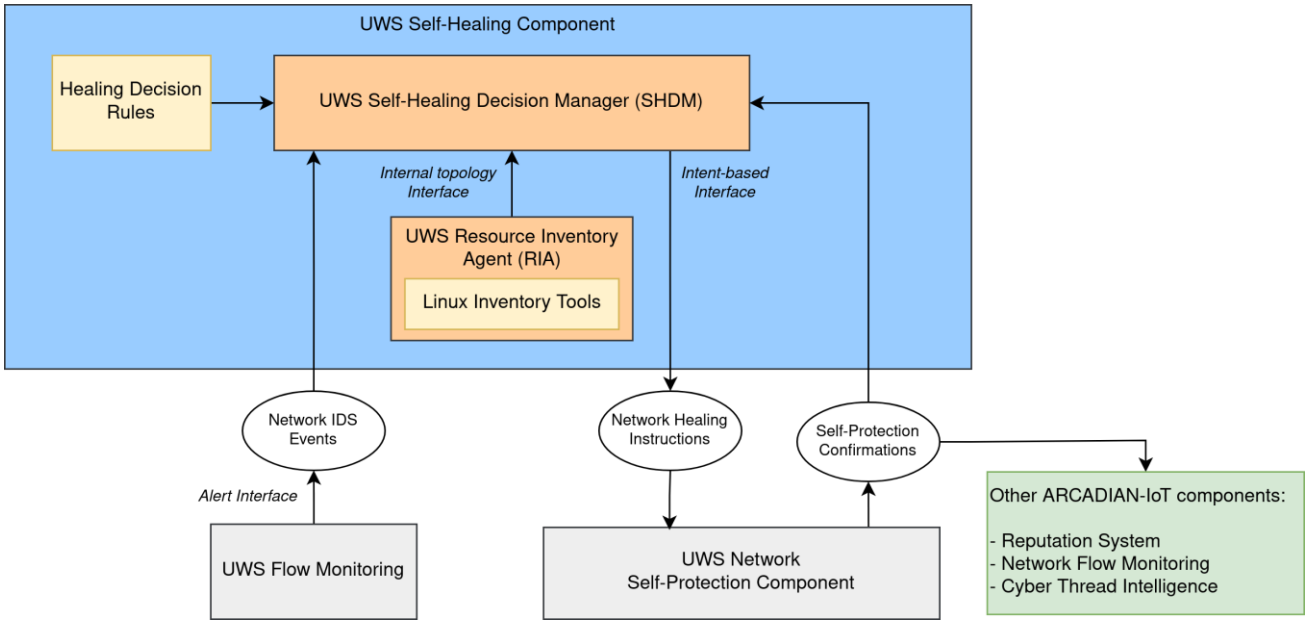

Figure 25 - Self-Healing Architecture Model

Internal interfaces

- The SHDM has as an internal southbound API for the information received from the RIA about the topological information discovered;
- The RIA has a northbound API to report all the devices, device ports and inter-connections discovered in the infrastructure. The interface will use AMQP 0.9.1 protocol to send the results of the detection using a JSON format to a publication/subscription middleware such as RabbitMQ or similar;
- The RIA has a southbound API implemented as an internal execution of protocols, daemons and user space Linux tools to perform the inventory of the network topology. (e.g., LLDP, CDP, SMTP, iproute2 tools, etc).

External interfaces:

- The Self-Healing Decision Manager (SHDM) has as a southbound API for the information received from the Network IDS Events where the northbound interface of Network Flow Monitoring is reporting;
- The SHDM has as a southbound API for the Intent-based API where the northbound interface of the Self-Protection Component is subscribed.

3.5.5 Network Self-protection

This component oversees the network self-protection capabilities provided by the ARCADIAN-IoT framework. The component, whose architecture is depicted in Figure 26, is responsible for enforcing protection rules into the data plane and thus, protecting the infrastructure, IoT devices and services against attacks. To achieve this purpose, the component is hooked into the data plane, and it provides an enabler to enforce protection rules into the data plane. These capabilities are provided by the UWS OpenVSwitch Datapath Security Controller. This implementation is based on the well-known software OpenVSwitch which will be extended to provide support for protection rules into IoT networks, overlay networks, and any other ARCADIAN-IoT related networks.

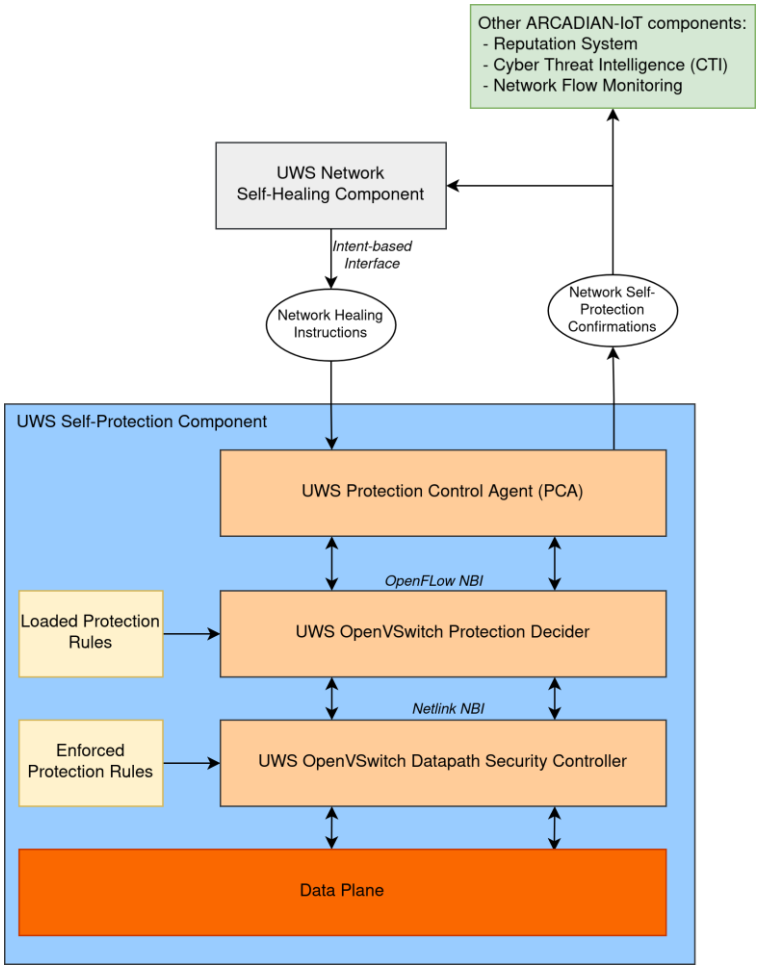

Figure 26 - Network Self-Protection Architecture Model

The self-management capabilities are achieved by the UWS OpenVSwitch Protection Decider which is in charge of deciding on every instant what subset of rules from the complete set of protection rules located into the “UWS OpenVSwitch Protection Decider” to be enforced in the “UWS OpenVswitch Datapath Security Controller” creating an autonomous control loop to perform self-optimization of the protectional capabilities and thus achieving large-scalability, really needed to deal with security on IoT networks. Finally, the UWS Protection Control Agent (PCA) will be oversee interfacing the internal components in order to perform the installation, enforcing of the protection rules into the self-management system. UWS’s PCA will expose an intent-based external interface that will receive technology-independent instructions that will be translated into technology dependant commands to be enforced in the data plane.

Internal interfaces

- OpenVSwitch Security Controller provides a southbound API that allows to enforce protection rules into the data plane. UWS’s OpenVSwitch Security Controller will be implemented as a kernel module where the enforced protection rules will be loaded from UWS’s OpenVSwitch Protection Decider using the Netlink protocol. Netlink is the standardized protocol distributed with Linux to allow inter-process communication (IPC) between both kernel and user-space processes, and between different user-space processes, allowing them to send, receive and share information.
- OpenVSwitch Protection Decider has a southbound internal API making use of the Netlink protocol to enforce rules into the data plane by sending them to UWS’s OpenVSwitch Datapath Security Controller in kernel space.

- OpenVSwitch Protection Decider has a northbound internal API making use of an extended version of the OpenFlow protocol provided by ARCADIAN-IoT to allow fine-grained control of IoT networks.
- OpenVSwitch Protection Decider has an east-west interface with a ruleset that is used to load all the protection rules that will be managed by the self-management system.
- PCA is using the northbound API of UWS's OpenVSwitch Protection Decider to perform the enforcing the whole set of the protection rules into the self-managed data plane.

External interfaces

- The northbound interface of the UWS's PCA) is an intent-based Interface exposing a control interface to enforce protection rules against attacks. The interface will use AMQP 0.9.1 protocol to receive intents from the Network Self-Healing component using a JSON format from a publication/subscription middleware such as RabbitMQ or similar. The main interaction of this component involves the Network Self-Healing component. This cooperation is aimed at providing, in conjunction with the Network Flow Monitoring component, a cognitive network self-healing loop that, in an automated way, with no need for human intervention, is able to detect, stop and mitigate know threats (DDoS attacks). As the communication between Network Self-Protection and Network Self-Healing is based on a subscription service, other ARCADIAN-IoT components such as the Reputation System or the CTI that may be also interested in the information contained in the Self-Protection confirmation messages can subscribe to this message exchange.

3.5.6 IoT Device Self-Protection

This component will be responsible for protection at the IoT device level. Its role will be to apply protection rules and policies that prevent intrusions on devices or malicious applications from running sensitive code that could compromise the data stored on that device.

To achieve this purpose, the IoT Device Self-Protection component will receive information about possible intrusions and threats detected by Behaviour Monitoring and CTI related to IoT devices. Based on this, the IoT device self-protection component will analyse the data in the policy enforcer, which will decide the best way to act and protect the system.

Once the system is protected and free from threats, the Device Self-Protection component must send a request to the Self-Recovery component to allow quick recovery of data and services after incidents.

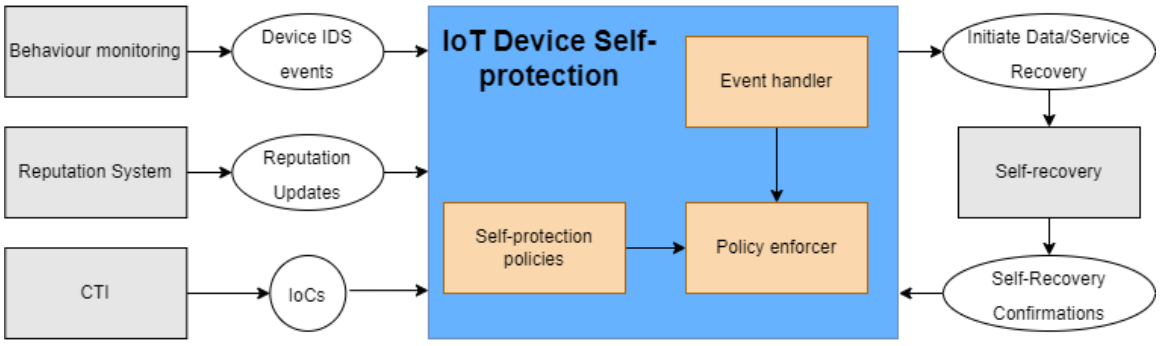

Figure 27 - Device Self-Protection Architecture Model

The eSIM (section 3.1.2) will contribute to the device self-protection by using reputation-based

authorization information (distributed by the authorization component – section 3.2.2) to act as an independent security agent within the device, able of performing protection actions in non-cooperative devices[13].

Internal interfaces:

- Event handler: Handles incoming events from other ARCADIAN-IoT components like Behaviour Monitoring component (e.g., threat / intrusion detection warnings), and CTI (e.g., Indicators of Compromise or external threat/intrusion-related information). It also translates and forwards the incoming information towards the internal policy enforcer sub-component.
- Self-protection policies: database with self-protection policies bootstrapped within the service (with envisioned support for remote policy updates). This sub-component serves as a knowledge base for the policy enforcer.
- Policy enforcer: It oversees applying/enforcing specific actions on the device (e.g., revoking permissions, close active connections, among others). Based on the incoming events info, the policy enforcer probes the self-protection policies database in order to apply the most appropriate actions according to the selected policies.

External interfaces:

- Behaviour Monitoring: Interface that receives intrusion information of the Behaviour monitoring component.
- CTI: Interface used to receive information (Indicators of Compromise) from the CTI component.
- Self-Recovery: Interface to inform self-recovery component to recover data and/or services after incidents.
- Reputation System: Interface to consume reputation score updates (e.g., used by the policy enforcer to infer the most appropriate policies also taking into account device reputation).

3.6 Common plane

The common plane is composed of 2 main components:

- Hardened Encryption, which is supported via 2 different RoT approaches: 1) the eSIM (led by XLAB), 2) the Cryptochip (led by BOX2M);
- Permissioned Blockchain (led by ATOS)

3.6.1 Hardened Encryption

ARCADIAN-IoT's Hardened Encryption (HE) aims at providing a system that is more flexible, decentralized and further hardened by two main RoT hardware options: the hardware-based root of trust (RoT) provided by the eSIM component, or the hardware-based root of trust (RoT) provided by a cryptochip (which may either be embedded or an add-on module integrated by a

[13] Details removed for potential IPR protection purposes

vendor into its existing IoT device).

3.6.1.1 Hardware-based encryption supported by eSIM as RoT

The hardened encryption component, depicted in

Figure 28, will enable protecting various types of data in ARCADIAN-IoT, by providing modern encryption mechanisms further hardened by a hardware-based root of trust. The component tackles three main challenges:

- Key and policy management of a large data producing system by introducing Attribute-Based Encryption (ABE) mechanisms. ABE aims to overcome traditional encryption in which shared keys need to be established between any data producer and receiver. In contrast, ABE allows encrypting data according to given policies that desired decryptors must satisfy. This allows more flexibility, simplifies the management, and drastically decreases the number of keys in the system that could be vulnerable to attacks.
- Removing a single point of failure by decentralizing the authorities managing the access to keys. The hardened encryption component will provide a decentralized key management solution that will together with blockchain enhance the reliability of the system.
- Establish further trust in the source of the data by providing signatures of the encrypted payloads from a hardware-based root of trust (e.g., eSIM - section 3.1.2). This will allow the receivers of data to verify the source.

Two major subcomponents will be provided with this approach: a library that can be used by the entities in ARCADIAN-IoT to encrypt, decrypt and sign data, and (decentralized) key management servers.

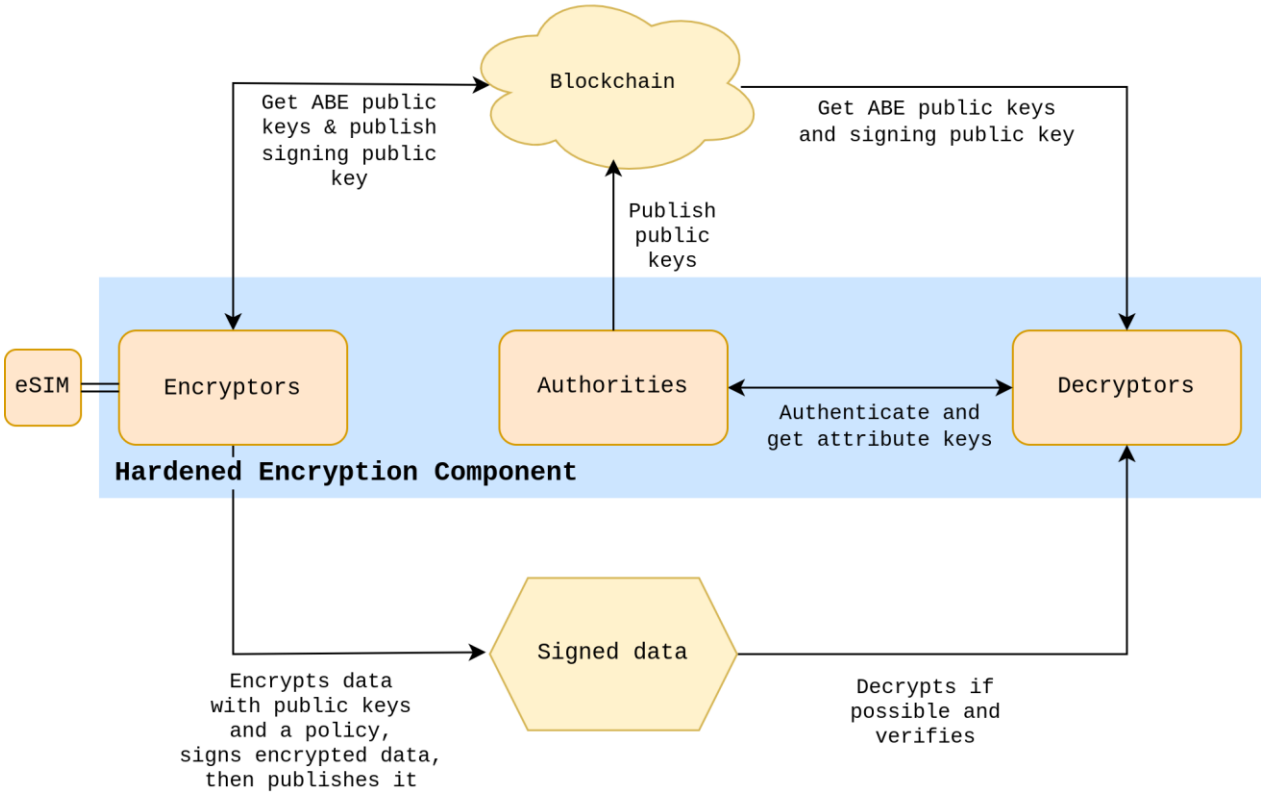

Figure 28 - Hardened Encryption Architecture Model

Internal interfaces:

- Key generation interface: devices, users and other entities will be requesting and receiving cryptographic material enabling them to participate in Arcadian-IoT platform.

External interfaces:

- eSIM component interface will enable signing an encrypted payload or a communication session on the request of the hardened encryption component present in the device;
- Self-aware data privacy component will provide policies that the devices and users can use to anonymise or encrypt/decrypt their data;
- Decentralized identifiers and permissioned blockchain will allow identification and authorization to key management servers as well as decentralized key storage.

3.6.1.2 Hardened Encryption supported by cryptochip as RoT

The hardened encryption system leveraging cryptochip-based RoT is aimed at scenarios requiring secure end-to-end sensors data traffic transfer to/from the IoT management platforms.

This system is composed by 2 main components: the device firmware and the middleware application. They will provide protection to various types of industrial telemetry data in ARCADIAN-IoT, by employing modern and disruptive encryption mechanisms further hardened by the cryptochip as hardware-based root of trust. The encryption system addresses several technological challenges:

- Lack of capability of microcontroller powered IoT devices used in industrial field to support / run TLS encryption, due to lack of computing power;
- Embedded crypto chips into such devices, if there are, require and are completely dependent by a local / on-premises software application and IT infrastructure for key handling;
- Lack of capability of existing key handling applications to provide also a relay encryption function, in relation with other applications, requiring traffic data from sensors interfaced by IoT devices; usually, traffic ends into this application and is not forwarded safely to a data management platform;
- Current IoT platforms are less / not familiar with handling data encrypted by non-TLS methods, and cannot handle hardware encrypted devices;
- There is a high demand from grid industry for such security optimization, on top of ENISA requirements and considering existing deployed infrastructure (expensive, closed, less integrable) of field monitoring equipment;

The hardened encryption system, mainly targeting secure end-to-end communication in grid sensors infrastructures, requires the adoption of a specific middleware application running the following services:

- a **key management system** (such as the one applied in the eSIM-based RoT approach),
- an agent for managing the cooperation with other ARCADIAN-IoT services (e.g. Reputation System, Device Behaviour Monitoring)
- a **set of interfaces to manage the interaction with 3PP** systems and decentralized authorisation mechanisms (which may be provided by ARCADIAN-IoT or by 3rd party Providers)

The system is designed to be used agnostically by external / other IoT platforms and devices, by technology developers willing to integrate it into their ecosystem, for both retrofits and greenfield

developments.

The device can operate with both regular SIMs and eSIMs (for pure connectivity and ARCADIAN-IoT compliant authentication purposes, not for encryption purposes, these being in the role of the embedded crypto chip), across multiple mobile networking technologies, but also with fixed connectivity (ETH and WiFi ports).

The firmware of the IoT device, containing the crypto chip, provides:

- an agent for handling the crypto keys: embedded into IoT device firmware, will follow the data formats and specifications required by the crypto chip supplying vendor. It will be responsible for extracting and validating the necessary keys for each of the steps in the authorization process and/or in telemetry (grid sensors) traffic management, as well as providing the primitives for running the encryption and decryption functions in the firmware core system.
- an agent for handling decentralized authorization, which will integrate a purpose-fit opened source protocol (such as DIDComm), and will be embedded into the IoT device's firmware, respecting protocol specifications.

Internal interfaces:

- **From the device point of view:**
  - interface between encryption agent and encryption local repository (for storing assigned keys);
  - interface between encryption agent and sensors data sources;
  - interface between device firmware frontend and Self-Recovery through a terminal application.
- **From the middleware point of view:**
  - interface between encryption / decryption service and encryption DB (storing all devices keys);
  - functional interfaces of telemetry service, its database and HTTPS/TLS engines (internal) with encryption / decryption service.

External interfaces:

- **From the device point of view:**
  - interface between decentralised authorisation agent and decentralised authorisation system
- **From the middleware point of view:**
  - functional interfaces of telemetry and encryption system, their databases and HTTPS/TLS engines (external) with systems developed under ARCADIAN-IoT scope (DIDs / decentralised authorisation or Reputation System[14]).

Figure 29 depicts internal and external interfaces between the IoT device and the middleware platform (containing the encryption & decryption system, i.e. the Hardened Encryption

[14] Currently well-defined interactions with Hardened Encryption supported by Crypto-chip RoT. The potential interface with other ARCADIAN-IoT components is subject to developments in WP3

functionality), respectively with some potential ARCADIAN-IoT components (e.g. Self-Recovery).

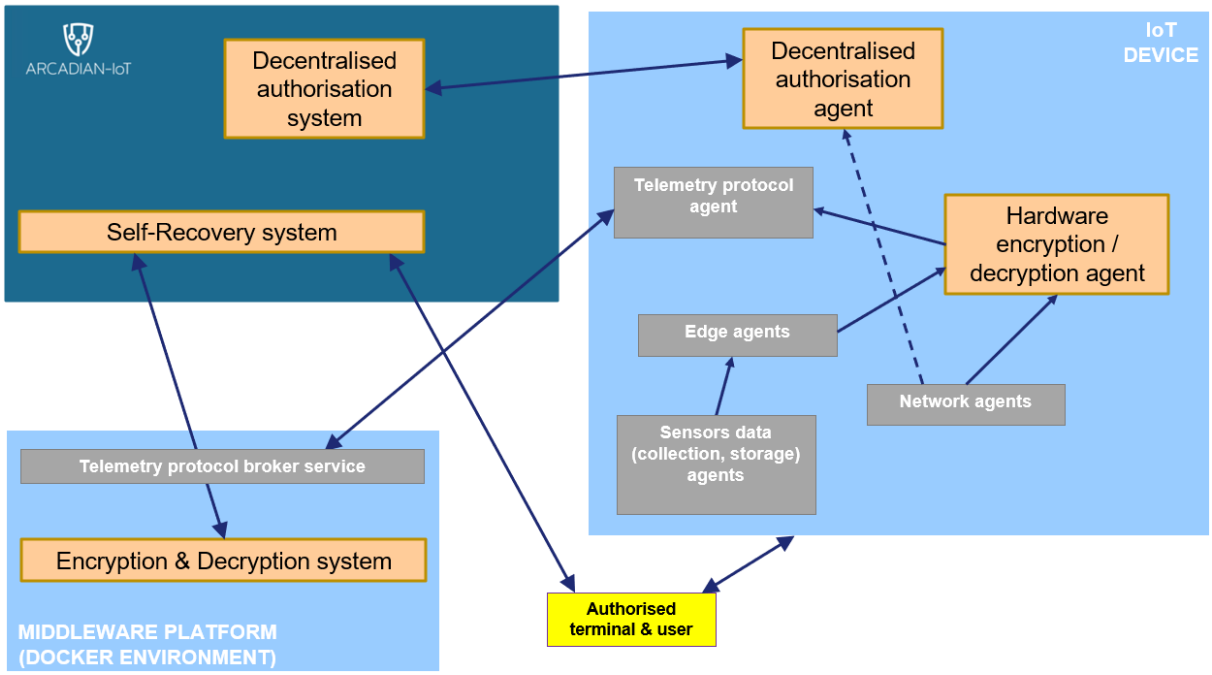

Figure 29 - Device Firmware and Middleware Interfaces

Figure 30 visually describes internal and external interfaces between the middleware platform (containing the ARCADIAN-IoT encryption system) with the other platforms and ARCADIAN-IoT systems developed by other partners.

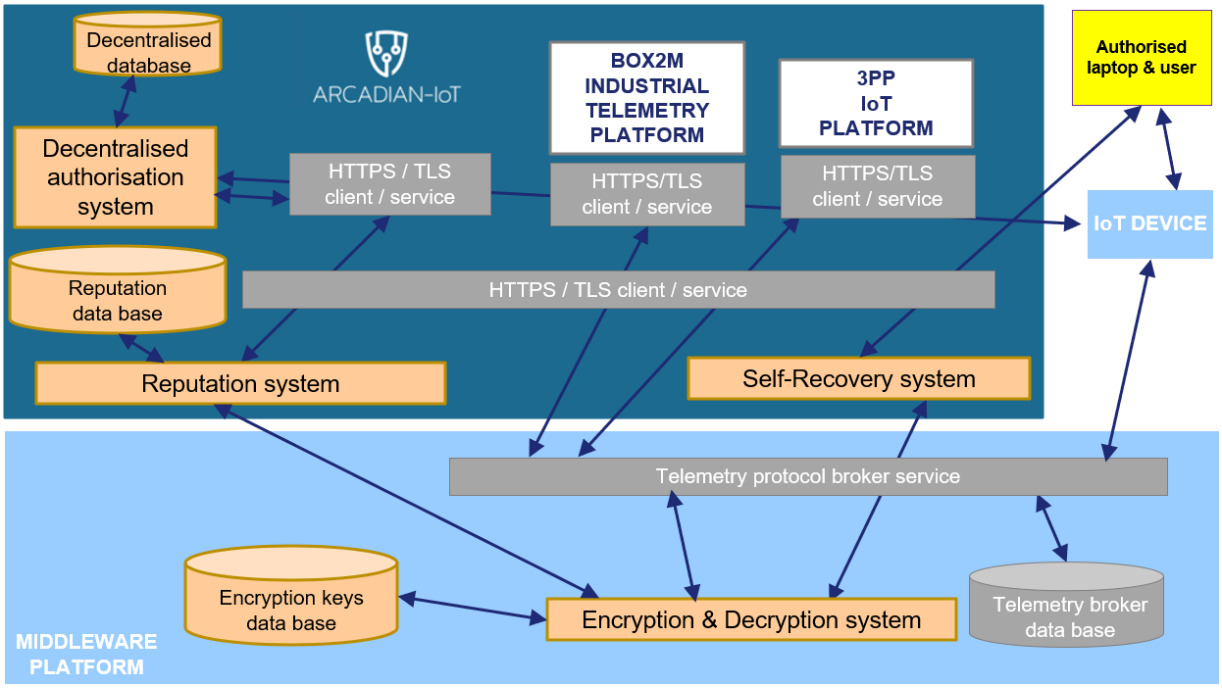

Figure 30 - Middleware Interfaces with other ARCADIAN-IoT internal and external services

An important note regarding the presented figures: considering BOX2M's dual role, i.e. as technology researcher and domain owner, the provided diagrams include additional information relating to prototyping plans. The following information is provided to support visual

interpretation:

- orange boxes represent ARCADIAN-IoT components to be developed across the project;
- grey, white and yellow boxes represent external and existing components (which are already expected to be used as enablers for prototyping and deployment):
    - grey boxes: represents existing services (BI modules, applications) used for data transfer;
    - white boxes: represents IoT platforms (BOX2M and other vendor) used to end-point the encrypted data. A terminal from which the Recovery is accessed is also represented.

3.6.2 Permissioned blockchain

The permissioned blockchain is one of the horizontal components that will be part of the ARCADIAN-IoT framework serving vertical components such as identity management, that will make use of the blockchain´s immutable auditability and traceability properties. Given the sensitive nature of data that will be shared in the network, ARCADIAN-IoT will use a private permissioned blockchain approach.

Private permissioned blockchains place restrictions on who is allowed to participate in the network and in what transactions. It can have several conditional access features for users to obtain permission to operate at given levels. In order to interact with the blockchain, software clients typically run their own node, automatically updating the common state internally and then notifying the rest of nodes. Importantly the permissioned blockchain has much higher throughput and aimed more at enterprise solutions whereas the public permissionless blockchains are much slower due to their global participation and complex consensus proofs [3].

Figure 9 shows a high-level example of a blockchain network with three nodes that are envisaged to be deployed and hosted by pilot partners.

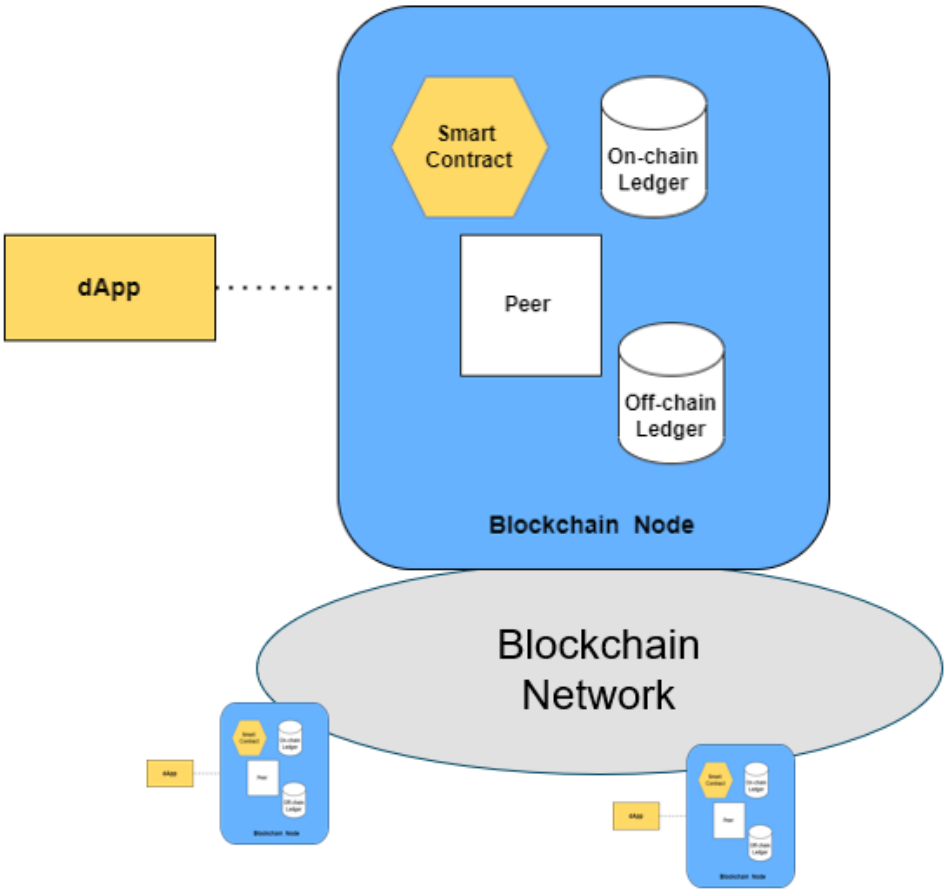

Figure 31. Permissioned Blockchain Architecture

Decentralised client application (dApp) proposes read / write transactions to the smart contract. Smart Contracts receive the client transaction proposals and agree to endorse or not the result to read or write to the blockchain. On-Chain ledger holds a replica of all data stored in the blockchain network of peer nodes. Off-chain ledger holds private data with the integrity provided by on-chain hash function, but that this data can be deleted without any trace and can also be shared to 3rd parties with its integrity checked against the on-chain blockchain.

Internal interfaces:

- On-chain ledger interface
- Off-chain ledger interface

External interfaces:

- Decentralised applications provide the transactional and event monitoring interfaces to the blockchain peer nodes hosting the ledger. Components that will use such apps in the vertical layer include:
  - Decentralized Identifiers
  - Reptation System
  - Hardened Encryption
- Wallet interface proves the identity of the user
- Oracle interface for obtaining external input

The following section introduces ARCADIAN-IoT main data flows, communication solutions, and respective validation domains.

4 ARCADIAN-IOT INFORMATION VIEW AND VALIDATION DOMAINS

This section starts by addressing (Section 4.1) the different main flows of information, that are envisioned within ARCADIAN-IoT framework, and some preliminary details (e.g. data communication and storage choices). Furthermore, the foreseen positioning of ARCADIAN-IoT components and the service's trust boundaries for each of the three validation domains (i.e., Domain A, B, and C introduced in Section 2.1) is presented (Section 4.2).

4.1 Information View

There are diverse types of data being exchanged between ARCADIAN-IoT components. The information exchange takes place via different communication mechanisms, each being best suited for each situation. This section presents the ARCADIAN-IoT data flows and preliminary specifications regarding the supporting communication mechanisms (i.e., message bus and REST APIs) and storage approaches (e.g. such as the permissioned blockchain).

4.1.1 ARCADIAN-IoT Data Flows

ARCADIAN-IoT Data Flows represent the type of information that is exchanged between the architecture components. This exchange can take place in the form of one-to-one or on-to-many (as previously described in Section 2.3). The data flows that are currently defined – but might be subject to change as the result of research and development activities, which are still in its initial stages - are the following:

- An **Indicator of Compromise (IoC)** is a piece of data containing technical and non-technical information about threats and attacks observed in the network. In ARCADIAN-IoT, IoCs provided by external entities are aggregated and filtered by the CTI. The CTI will also gather local threat data to generate and share IoT-specific IoCs in a uniform format.
- **Device Intrusion Detection System (IDS) events**, pertains to the outcome of the analysed sequence of events related to the IoT device at hand by the Behaviour Monitoring component. This outcome will comprise the result of the classification made by the AI trained models, which can be either normal or intrusive behaviour along with a certain level of confidence of the prediction. In addition, if an intrusion was detected on the system, the Behaviour Monitoring component will be able to give the classification of the attack based on the already known attacks.
- **Network Intrusion Detection System (IDS) events**, refer to the data flow that the Network Flow Monitoring is publishing to the corresponding subscription/publishing exchange. This information contains a set of metrics related to the detected malicious flow, such as IP addresses (inners and outers), origin and destination ports, tunnelling IDs, a unique identifier of the malicious flow; information about the rule that has raised the alert (alert name, type, priority, reason ID); information about the component, such as the component's unique identifier.
- **Network healing instructions**, refer to the information that the Self-healing component is publishing to the corresponding subscription/publishing exchange, containing the result of its prescriptive analysis. It specifies WHAT action should be taken, WHERE this action should be enforced, WHEN it must be enforced and for HOW LONG it must be active, referring to the alert reported by Network Flow Monitoring component in the *Network IDS events* data flow.
- **Network self-protection confirmations** are ACK messages to inform of the result of the enforcement of a network healing instruction in the data plane, which are sent from the

Network Self-Protection component to the Self-Healing component. The self-protection confirmations are sent through the appropriate subscription/publishing exchange, being their main consumer the Self-Healing component. The messages are reported in JSON format and are composed of two sections:

  - The original network healing instruction generated and sent by the Self-Healing component because of a DDoS attack alert raised by the Network Flow Monitoring component.
  - The outcome of the enforcement of the network healing instruction in the data plane: a time stamp and the result (e.g., “Action enforced” or “Error”).

- **(Network) authorization** data flows are the following: (1) between the Reputation System and Self-Recovery with the network Authorization component, which form the basis to automatically create the new network-based policy control rules to be enforced there; (2) between the network Authorization component and the hardware secure element (eUICC/eSIM), particularly with ARCADIAN-IoT eSIM profile, to provide device trustworthiness information so it can take security measures locally.
- The reputation system supports two main types of data flows: the reputation updates/scores, and the reputation policy management. They can be defined as follows:
  - The **reputation updates/scores** refer to the data flows between the reputation system and the components subscribed to the topic of reputation updates, for instance the network Authorization component. This data flow type conveys information regarding the reputation score and the policies that can be applied considering the score provided.
  - the **reputation policy** management allows a domain owner to configure the desired policies according to the reputation scores. For instance, the reputation policies can include information, such as when the person reputation is below a certain threshold, a user is not able to access a certain service (for instance to register or use the Drone Guardian Angel (DGA) service in the domain A). The data flow refers to an API that is made available by the reputation system to specify and update existing policies. It should be noticed that these policies mainly affect the authorization component.
- **Authentication events** refer to credentials provisioning and verification, in a multi-factor authentication scheme, which counts with decentralized identifiers, network credentials and biometrics as authentication factors (and the related data flows). It also refers to the authentication results provided to the device behaviour monitoring component. Finally, it refers to authentication tokens, provided to the self-aware data privacy component, for role and privacy-based authorization enforcement.
- **Network credentials** flow refers to the data that comes from IoT devices hardware that allow its identification and verification in the core network, which, in case of successful identification, endorses a token for authentication in third party services (where the token is verified and validated within the authentication process).
- **Device attestation evidence** refers to the set of data that a device produces to prove its integrity as a form of measuring its trustworthiness. The evidence stems from claims which (at the device) will be encrypted for ensuring confidentiality, and signed via hardware-based root-of-trust for ensuring the sender integrity. Sets of claims (Evidence) will be validated by one or more Verification components according to their appraisal policies, with the associated result impacting device reputation.
- **Recovery data flows** refer to messages exchanged to enforce recovery of data in case of incidents, initially either manually, through Device Self-Protection (after identification of

concrete threat prone to recovery)

The inter-component communication solutions considered by ARCADIAN-IoT are presented next.

4.1.2 ARCADIAN-IoT Inter-Component Communication

ARCADIAN-IoT framework is composed of several individual components that form a Chain of Trust and enable security, privacy, trust, identity and recovery on IoT-driven domains. To support such inter-component communication, secure and reliable communication mechanisms will need to be employed. This section provides preliminary information regarding expected tooling and technologies for achieving this. At this stage, ARCADIAN-IoT is planning to rely on the usage of well-proven message bus solutions (e.g., Apache Kafka or RabbitMQ) and other specific communication approaches like RESTful APIs and DIDComm[15] messaging.

4.1.2.1 Message Bus

By deploying an ARCADIAN-IoT managed message bus, the consortium can assure the necessary functional and operational requirements are met. As the message bus is the main communication mechanism of several modules, it is important to ensure persistence but above all security and privacy. Therefore, exchange messages will need to be end-to-end encrypted and only accessible to the intended subscribers, for which components authentication will be critical. Furthermore, considering scenarios with hundreds or thousands of devices, it is also necessary to assure a performant message bus, capable of handling large amounts of messages or able to easily scale according to the expected workloads.

There are several ways of assessing the performance of message bus mechanisms (e.g., different message sizes, number of instances, number of topics, and many others). Therefore, to assess them fairly and neutrally, the evaluation of different solutions needs to have the same basis and validation approach. Since such evaluation is out of the scope of ARCADIAN-IoT, an external evaluation comparing 3 main solutions is used to determine the most appropriate option for ARCADIAN-IoT.
Table 2 shows a comparison of RabbitMQ, Apache Kafka and Pulsar provided by Confluent[16].
The evaluation is built upon the Open Messaging Benchmark Framework (OMB)[17] with updates on the Java version and updates of the versions of each solution. According to the evaluation results, Apache Kafka indicates the highest peak throughput values and the second lowest latency. On the other hand, RabbitMQ demonstrated the lowest peak throughput but the lowest latency. It is worth mentioning that the latency is assessed on an average 200MB/s throughput for Apache Kafka and Pulsar, and 30MB/s for RabbitMQ.

Table 2 - Comparison of message bus solutions

Solution	Peak Throughput	Latency	Message Order	Message Persistence
RabbitMQ	105 MB/s	1ms	Yes	Yes

[15] https://identity.foundation/didcomm-messaging/spec/
[16] https://www.confluent.io/blog/kafka-fastest-messaging-system/#throughput-results
[17] https://openmessaging.cloud/docs/benchmarks/

		(30MB/s)		
Apache Kafka	605 MB/s	5ms (200 MB/s)	Yes (indirectly)	Yes
Pulsar	305 MB/s	25ms (200MB/s)	No	Yes

The majority of ARCADIAN-IoT component researchers have indicated (in a survey presented in Section 5) that RabbitMQ is a suitable choice to support the communication of ARCADIAN-IoT components. Since integration activities officially start in month 13 (May 2022), the results presented in Table 2 and partners preferences will be considered when integration activities define the precise message bus – or buses - which will provide support to the framework.

Irrespective of the final decision, the chosen message bus(es) is expected to address at least the following minimum requirements:

- End-to-end encryption of all data flows;
- Distinction of message types (e.g., direct messages or Remote Procedure Call (RPC) – where applicable);
- Authentication of publishers and subscribers.

4.1.2.2 REST APIs and DIDComm

While several ARCADIAN-IoT components consider the usage of pub/sub mechanisms via a common message bus, components such as Self-Aware Data Privacy or SSI (with Decentralized Identifiers) rely on REST APIs and DID Comm messaging to communicate. Other components leverage a combination of the two approaches (e.g. Authentication component).

RESTful API's do not require a detailed introduction as they are available and widely used for over 20 years. Regardless of their maturity, it is always necessary to guarantee that implementations cover at least five aspects to be fully compliant with the design[18]:

- Assume a stateless behaviour;
- Support a client-server approach;
- Adopt uniform interfacing;
- Implement the ability to cache;
- Devise a modular design.

Besides RESTful APIs and the message bus, DIDComm messaging solutions are also considered for inter-component communication. DIDComm is a methodology whose purpose[19] is to provide secure and private communications over solutions with Decentralized Identifiers (DIDs).
As such, it is at this stage expected that ARCADIAN-IoT will support the 3 aforementioned communication mechanisms. Nonetheless, the employment of these additional communication mechanisms does not represent a significant overhead in the overall ARCADIAN-IoT Framework architecture. It is fundamentally a technical decision derived from multiple factors, such as the communication patterns and requirements of each component, as well as partner's own

[18] Roy T. Fielding and Richard N. Taylor. 2002. Principled design of the modern Web architecture. ACM Trans. Internet Technol. 2, 2 (May 2002), 115–150. DOI: https://doi.org/10.1145/514183.514185
[19] https://identity.foundation/didcomm-messaging/spec/

knowledge and previous experience.

4.1.3 ARCADIAN-IoT Permissioned Blockchain

Blockchain is an immutable and secure technology as it is only possible to append records and previous registries cannot be altered. Given that it is a decentralised technology, blockchain provides increased transparency and trust assurances, providing immutable auditability and traceability properties to the data under management.

In the context of ARCADIAN-IoT, there is a need of a repository to cooperate in the task of building Trust over the system. There are several kinds of repositories under consideration as SSI repository; even though Blockchain was one of the first options in SSI bibliography, other alternatives have been gaining attention, like DNSs and IPFS. All of them have strong and weak points. For instance, while DNSs would offer a more universal service it presents higher time rates to add information; on the other hand, IPFS seem to provide a more agile timing response but poorer immutable solution.

In the context of ARCADIAN-IoT, there is the need for a repository to cooperate in the task of building Trust over the system. There are several kinds of repositories under consideration as SSI repository, as even though Blockchain was one of the first options in SSI bibliography, nowadays other alternatives are gaining attention, like DNSs and IPFS. All of them have strong and weak points. For instance, while DNSs would offer a more universal service it presents higher time rates to add information, in the other hand IPFS seem to provide a more agile timing response but poorer immutable solution.

In a first analysis of the needs of ARCADIAN-IoT we envisage:

- There would be several types of information hosted in the repository. In the case of SSI, the information will be public cryptographic proofs of an identity which could be added by any SSI agent, while in other cases such as reputation it should be added only for specific entities. Therefore, it would require control over who is able to add information to the repository.
- In all scenarios under consideration for the repository, the information to be hosted could require a low time range to be added, I.e. a low latency. For example, the time between generating a reputation score and its publication should be short enough to let users take decisions with the proper data.
- Although the information to be hosted cannot be private data, because specific requirement of SSI paradigm, it would be the case that this information is of the interest of a specific group and not universally available to everyone, therefore the repository should provide access management. For example, the public proof of a DID can be public for everyone while the reputation score of a thing in a factory should only be accessible by the management of the factory.
- In all the scenarios we envisage the need of immutability, this is, although information could be updated, the repository must keep all the historic information. For example, in reputation context it will enable to build a history about the performance of a thing, while in SSI, although an agent will update their keys, already issued DIDs require to be proved later, like in the case of DID used to perform a purchase.

Having all these requirements in mind, although the different technologies; DNSs, IPFS, BC, etc. are able to provide a rival solution, it is considered that Blockchain presents the better match, considering also the specific use cases and requirements of ARCADIAN-IoT.

There are ARCADIAN-IoT components that rely on blockchain as a part of their functionalities.

The is the case of the reputation system, hardened encryption and SSI. A summary of the main reasons why such components rely on Blockchain is described next:

- The **Reputation System** will store reputation scores in the blockchain. The use of the permissioned blockchain, following a smart contract model is motivated with the need to store in a secure and reliable fashion the reputation scores. The purpose is to allow the storage of scores with the assurance that reputation information can be decentralized and distributed in a trustable fashion. The reputation system will store the reputation score in the blockchain, assuming that blockchain nodes are maintained by service providers, for instance, between the drones (from Domain A) and other providers that might be interested in the reputation values.
- The **Hardened Encryption** component will use blockchain as a reliable storage of public keys needed to encrypt data for specific recipients or to validate the authenticity of signatures. This enhances trust in the system due to the decentralized nature of blockchain. In fact, if the public keys are compromised, the privacy of the encryption can no longer be guaranteed. Hence the main benefit of the use of blockchain is preventing a single point of failure.
- The **Decentralized Identifiers** that support the Self-Sovereign Identity paradigm make use of blockchain in this architecture to anchor the trust of associated documents that publish public keys for authentication and verification of Verifiable Credentials.

Note that in ARCADIAN-IoT, the actual data of interest to each of the applications is not actually published on-chain but rather the blockchain is used as a notarization ledger to prove the integrity of the ARCADIAN-IoT application data stored off-chain.

The permissioned blockchain will restrict the organisations and actors that can write to the ledger, however depending on the use case application it may be that the data published on the ledger is available publicly.

The following section provides an overview of ARCADIAN-IoT Validation domains, assets, actors and trust boundaries.

4.2 ARCADIAN-IoT Validation Domains

This section reflects on the domains and respective services described in Deliverable 2.2 [2], addressing the information view while aiming to perform a first step in the threat modelling process. Performing a threat modelling analysis for each of the IoT solutions (DGA Service, Grid Management, Medical IoT) is critical to guide the security functionality of each ARCADIAN-IoT component and the framework itself. As such, for each domain, a high-level diagram depicting context-specific assets, actors and the associated trust boundaries is presented. As defined by OWASP[20], in the context of threat modelling, a trust boundary "*is a location on the data flow diagram where data changes its level of trust*". It is important to state that following a comprehensive threat modelling for each of the technical components is out of scope of ARCADIAN-IoT; nevertheless, such component-centric threat modelling, building on the business-centric threat modelling, may be subject to research according to the importance for fulfilling the project and specific component's objectives, and as such being addressed within WP3 (for components from the horizontal planes) and WP4 (for components from the vertical planes). Thus, while we currently disregard trust boundaries between ARCADIAN-IoT

[20] https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html

components, this will be enriched – not comprehensively addressed - based on future WP5 progresses, and via future developments in the research in the horizontal and verticals planes components of the architecture.

The following sub sections present the diagrams for each of the domains: (A) Emergency and Vigilance; (B) Grid Infrastructure Monitoring; and (C) Medical IoT.

4.2.1 Domain A

Domain A – Emergency and vigilance using drones and IoT, which can be considered as fitting a Smart City context, explores the usage of IoT devices (drones, more precisely) for enabling a private urban vigilance service for persons, which, upon registration, are able to initiate it via their smartphone.

As shown in Figure 32, within domain A there are five key elements. They can be described as follows:

- A **user** of the service, which interacts with the service for registration, initiation and usage purposes, may alternatively be a malicious actor intending to, for instance, intrude the device or abuse the service;
- The user's **smartphone**, which comprises:
  - Drone Guard Angel (DGA) app, implementing the service logic (e.g., enabling user registration, service activation, etc);
  - On-device ARCADIAN-IoT cybersecurity-related software (e.g., Behaviour Monitoring) and hardware-based (e.g., eSIM) functionality;
- Drone Guard Angel (DGA) **service provider**, responsible for providing DGA service complying with ARCADIAN-IoT framework (e.g., DGA service provider with reputation above a given threshold);
- DGA **drone**, which embeds ARCADIAN-IoT security, trust and recovery capabilities;
- Backend **ARCADIAN-IoT services**, running in the network or Cloud environment.

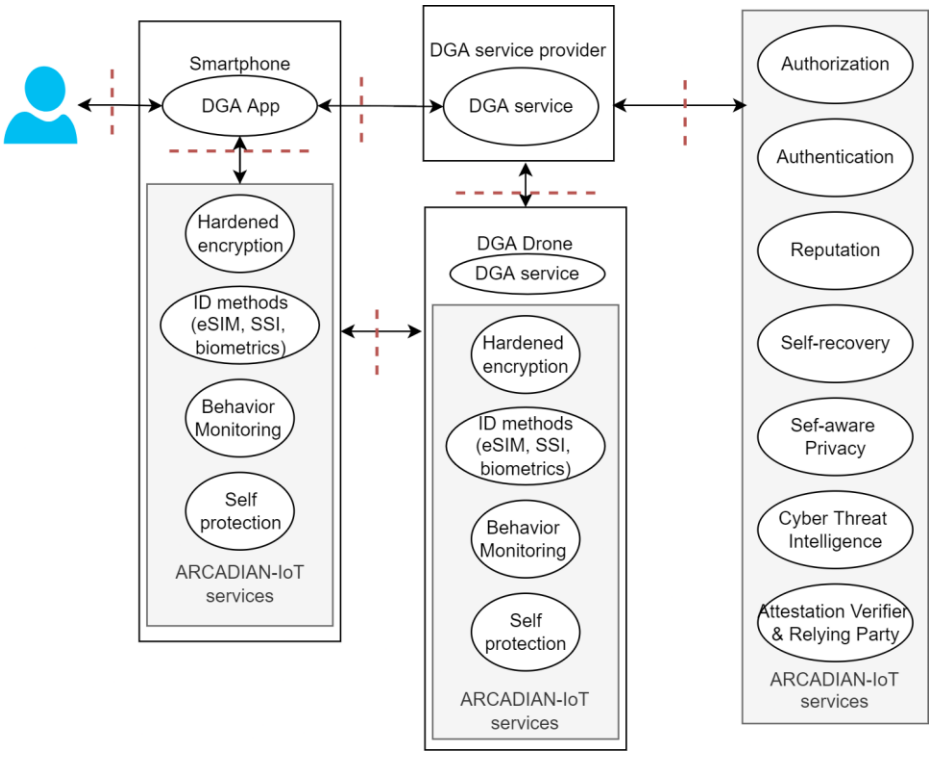

Figure 32 - Component Placement and Trust Boundaries Diagram (Domain A)

Figure 32 equally shows that the following trust boundaries (represented with dashed red lines) can be identified:

- User – Smartphone;
- Smartphone – DGA service provider;
- Smartphone – DGA drone;
- DGA drone – DGA service provider;
- DGA service provider – ARCADIAN-IoT services;
- (Within the smartphone) DGA Application – ARCADIAN-IoT services.

4.2.2 Domain B

Domain B - Secured early monitoring of grid infrastructures, considers an IoT solution for monitoring grid infrastructures. This solution monitors, for instance, the status, behaviour, consumption, production, quality of power or utilities infrastructure elements (e.g., power generators, substations, distribution, consuming gear). The solution enables grid managers to obtain an accurate perception of the grid operation and potential issues, and to intervene proactively and reactively for maintenance and optimization, using secured data.

As shown in Figure 33, within domain B there are six key elements. They can be described as follows:

- **Grid sensors**, actually providing the grid metering and monitoring (e.g., power meters, power analysers, debit meters, pressure sensors, environment sensors);
- **Grid actuators**, providing controlled changes into grid infrastructure (e.g., contactors, interrupters, actuators, controlling points);
- **GMS device**, an IoT device acting as gateway to the local grid sensors, doing cognitive datal collection from these, running edge computing on gathered data, encrypting these (via Hardened Encryption), and transmitting these data to the IoT platforms, through GMS middleware; oppositely, when receiving data, it runs decryption and forwards, through local edge engine, the execution of commands (OTA);
- **GMS middleware**, providing the collected grid data to end users (usually IoT platforms) via web services, and directly supported via a set of ARCADIAN-IoT services (e.g., Reputation System for storing device-related reputation information, Behaviour Monitoring, Recovery, services, Authorisation), as well as web services interfacing with IoT platforms and other external services – such as decentralised authorisation service);
- **GMS application user**, which interacts with the application for different purposes according to its role and associated levels of authorization (through Self-Aware Data Privacy). These may include e.g., Grid Infrastructure manager, external auditor, authorised recovery specialist / entity for correspondent situations, technology vendor using ARCADIAN-IoT technology as a service for its devices / IoT platform / both, proprietary vendor – BOX2M Engineering – for handling provisioning of new devices / changes on existing devices on authorised customer demand;
- **Backend ARCADIAN-IoT services**, running in private / public / hybrid Cloud environments, through various network connectivity technologies, providing additional services (e.g., Authentication, CTI), IoT data management – as monitoring, reporting, alerting, OTA provisioning & commands, web services with 3rd Party Providers (3PP)).

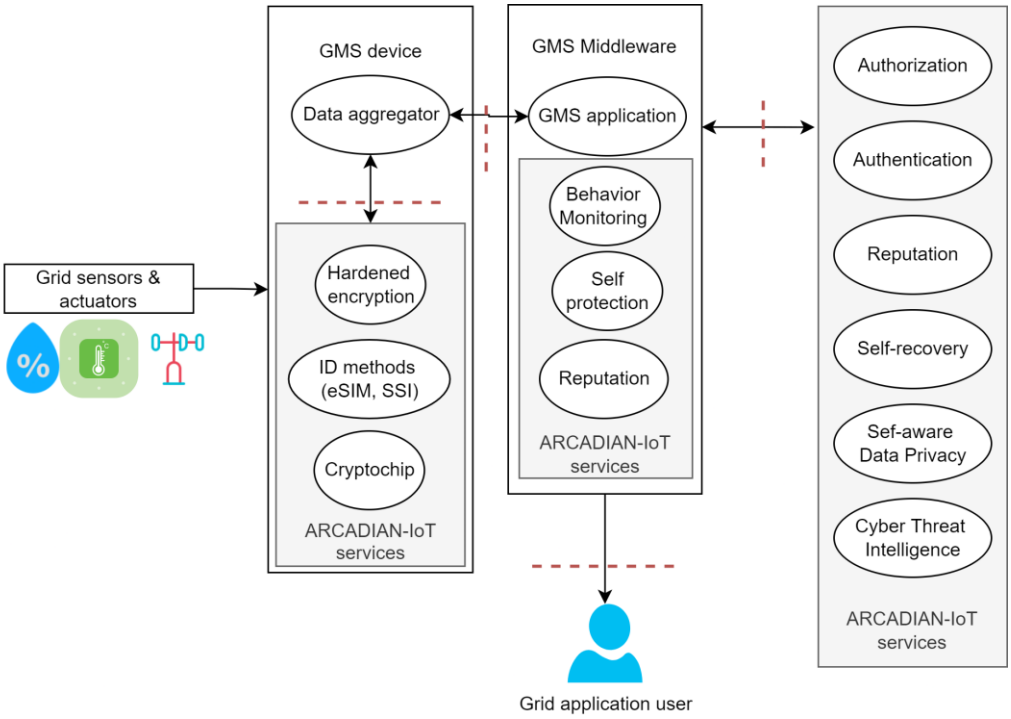

Figure 33 - Component Placement and Trust Boundaries Diagram (Domain B)

Figure 33 also shows that the following trust boundaries (represented with dashed red lines) can be identified:

- GMS device – GMS middleware;
- GMS middleware – backend ARCADIAN-IoT services;
- (Within the GMS device) Data aggregation function – ARCADIAN-IoT services.

Considering the GMS device will be running in a physically secured location, the interface between grid sensors and the GMS device is not considered a noteworthy trust boundary. Providing additional security mechanisms in the up to the GMS device is out of the scope of ARCADIAN-IoT and it is usually part of end customer physical security policies (within premises, resources, technology infrastructure). Anyhow, for safety purposes, BOX2M devices limited the type of connected sensors, filtering actively by its adaptive firmware the connected sensors, directly or indirectly, wiring physical connection only.

4.2.3 Domain C

Domain C – Medical IoT addresses a solution for remotely monitoring, via body sensor networks, health parameters for paediatrics radio-oncology treatments, enabling a valuable increase in comfort of targeted patients. Assuring secure data access (i.e., not being accessible by persons other than the medical staff) and integrity (e.g., in case of modification of obtained measurements through a malicious actor) is critical and has great relevance from the points of view of health treatment and privacy.

As shown in Figure 34, within domain C there are five key elements. They can be described as follows:

- Patient or **MIoT user**, which health parameters will be monitored;
- **Medical staff** which use the MIoT service in order to monitor the collected health data;
- **Medical IoT kit** comprising both the body sensors and the smartphone running the

medical IoT application as well as ;

- **Medical IoT service backend**, which comprises the middleware for securely distributing the collected health data and the monitoring tool for usage by medical staff;
- **ARCADIAN-IoT services,** providing all remote security, trust, identity and recovery management functionalities required for the Medical IoT service achieve the targeted trustworthiness.

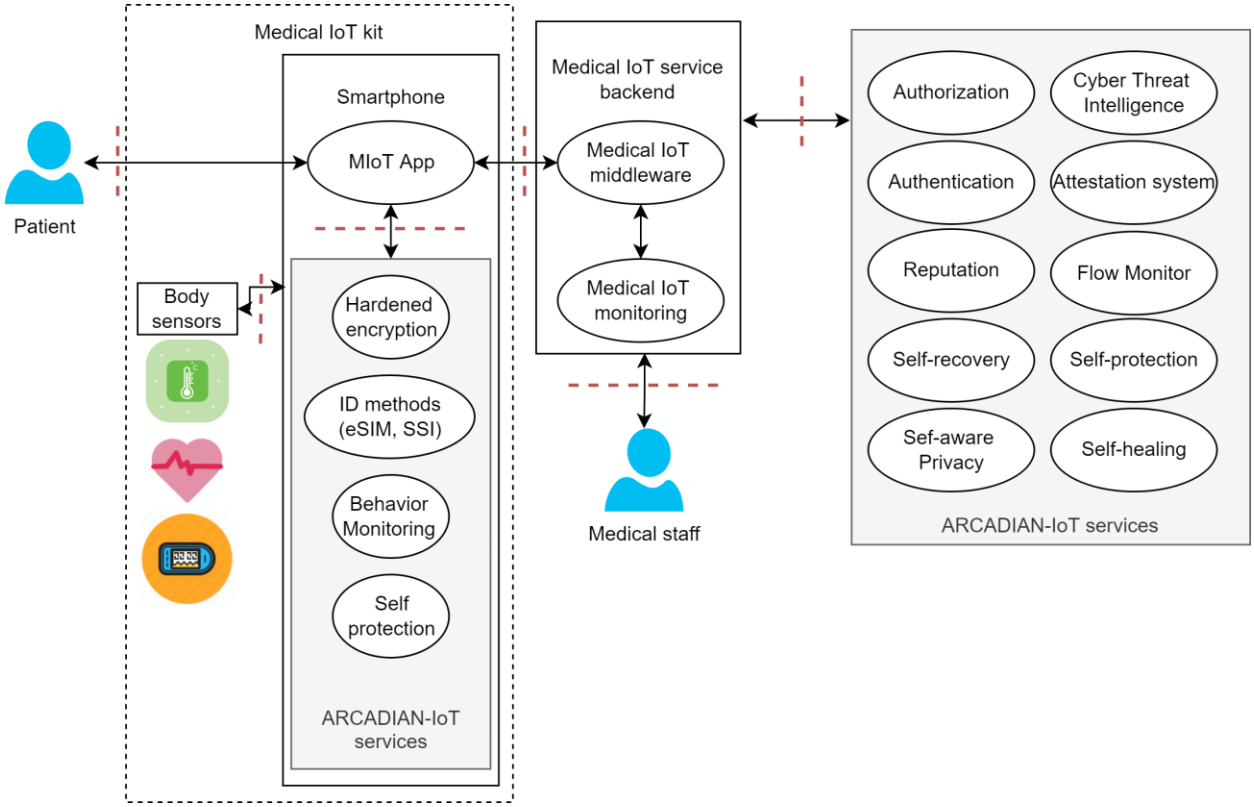

Figure 34 - Component Placement and Trust Boundaries Diagram (Domain C)

The figure also shows that the following trust boundaries (represented with dashed red lines) can be identified:

- Patient – Medical IoT Kit;
- Medical IoT kit – Medical IoT service backend;
- Medical Staff – Medical IoT service backend;
- Medical IoT service – ARCADIAN-IoT;
- Within the Medical IoT Kit:
  - Body Sensors – Smartphone
  - (Within the Smartphone) Medical IoT App – ARCADIAN-IoT services

The context-specific assets depicted in the previous sub-sections provide the reader with an overview of the actors and associated IoT solution-dependent trust boundaries within each domain. This initial threat modeling and trust boundaries analysis will be evolved aiming to identify key threats in the scope of planning ARCADIAN-IoT framework integration (WP5), and feed the research in the horizontal and verticals planes components of the architecture – where technical answers / solutions to such threats will be refined.

The following section presents the development and deployment views of the ARCADIAN-IoT framework, oriented towards timely preparing the integration efforts. In addition to describing the

technical development preferences and choices evidenced by ARCADIAN-IoT consortium, which will enable streamlining some integration efforts and decisions, it also presents a preliminary deployment view, which provides a notion of targeted distribution and positioning across the main different physical locations (i.e. device, communication network and Internet).

5 ARCADIAN-IOT DEVELOPMENT AND DEPLOYMENT VIEWS

This section addresses the development and deployment aspects of ARCADIAN-IoT framework. The main technologies and development processes of ARCADIAN-IoT components, as well as the underlying infrastructural considerations are described in section 5.1. On the other hand, aspects such as the targeted operating systems or deployment locations are presented in section 5.2. These are the result of a survey that was formulated in *EU Survey*[21] to collect development and deployment preferences among all technical partners. The survey collected feedback concerning 20 ARCADIAN components (i.e. 20 different forms were uploaded by partners). Therefore, there are 20 individual answers for each of the components. Following, we describe the main questions and respective answers:

It should be noted that, both considering the project R&I scope and its current stage, the provided development- and deployment centric information reflects initial indicators and partners expectations – and not decisions -, aiming to provide some guidance towards planning integration activities and reach the targeted TRL. Therefore, the information described in this section is expected to be subject to changes as result from final design decisions, development and integration activities.

5.1 Development View

Based on the answers collected from the survey, it was possible to conclude several aspects that affect not only the development of the individual components but also their integration within ARCADIAN-IoT framework. Namely, the preferred development programming languages, target operating systems, deployment locations, type of software packaging and several other, described later in this section.

- **Does the component depend on multiple sub-components?**
  Several ARCADIAN-IoT components rely on multiple sub-components to perform their functionalities. Table 3 shows that 80% of ARCADIAN-IoT components rely on sub-components. This is relevant as it provides additional insight on the structure of the components - revealing potential for not only portability but also increased flexibility to potential upgrades or targeted modifications on specific aspects of each component. Further details are provided in the technical deliverables that focus on the components' development (D3.1 and D4.1).

Table 3 - Sub-component dependence

		Answers	Ratio
Yes		16	80%
No		4	20%

- **Preferred programming language for development**
  Table 4 depicts partners' programming languages preferences. In questions where more than 2 options could be selected, the received input represents the likeliness or preference for a component to be implemented using a given approach (programming

[21] https://ec.europa.eu/eusurvey/

language, in this case). The results show the overall preferred programming languages for component development are GO, Java and Python. Every partner is able to develop in their preferred programming language as long as it can assure integration with other overall ARCADIAN-IoT framework, via its message bus or REST APIs. Other programming languages include bash, JavaScript, node, kotlin or rust. This is a pertinent aspect as the somewhat reduced scope of programming languages greatly facilitates cooperation among researchers and potentially enables easier development and integration activities.

Table 4 - Preferred programming languages

		Answers	Ratio
C/C++		2	10%
C#		0	0%
Go		10	50%
Java		10	50%
.NET		0	0%
Python		10	50%
Ruby		0	0%
PHP		0	0%
Other		4	20%

- **Does the component require a database?**
  An important aspect is the assessment of storage needs within ARCADIAN-IoT. As Table 5 shows, most components require storage. This may influence not only the development process, but also the deployment due to the software and hardware specifications (which are limited in IoT devices). The component that does not require database is the federated AI component. This is a natural outcome as this component provides a set of tools and mechanisms that complement the Federated AI capabilities of other components.

Table 5 - Database Requirement

		Answers	Ratio
Yes		19	95%
No		1	5%

The following questions provide further details on this matter.

- **What kind of database?**
  In this case, a set of popular database implementations were provided as options. Table 6 shows that Relational Database Management System (RDMS) or key-value stores are the most suitable (preferred at this point) solutions for most components. Nevertheless, as should be expected, other approaches will need to leverage other alternative, namely

blockchain or distinct file and object storage.

Table 6 - Types of Databases

		Answers	Likelihood
RDMS		10	50%
Key-value store		8	40%
Graph Database		0	0%
Other		5	25%

- **Where is the database going to be deployed?**
  Table 7 shows that most storage needs are addressed within the components. Therefore, there no significant impact or evident need of shared storage approaches within the ARCADIAN-IoT architecture. Only the decentralized identifiers and verifiable credentials (via SSI), Reputation System and Hardened Encryption require external storage (i.e., not within the component) which is likely to be supported by blockchain (also part of ARCADIAN-IoT framework), thus not impacting the global architecture of ARCADIAN-IoT. The blockchain support is a fundamental aspect of ARCADIAN-IoT Framework as it further enhances trust assumptions.

Table 7 - Database deployment locations

		Answers	Ratio
Within the component (or its sub-components)		16	80%
Externally		3	15%
No Answer		2	10%

The answers presented above show that there is great degree of flexibility within ARCADIAN-IoT component development approach. At the same time, this flexibility does not suggest, now, potential integration issues and incompatibilities.

ARCADIAN-IoT consortium partners indicated that at least 3 components (Self-Aware Data Privacy, CTI and Federated Learning) will be released in **open source** without restrictions. The partners further indicated that another 3 components (Device Behaviour Monitoring, Device Self-protection, and Hardened Encryption) will have partially open-source releases. The remaining components will not have open-source code available due to Intellectual Property Rights (IPR) concerns as well as other commercial exploitation strategies.

The solution that ARCADIAN-IoT adopted as a **code repository** is GitLab. Gitlab supports code versioning functionalities that are essential for the development of project of this scale. Additionally, GitLab also allows partners to employ **CI/CD** pipelines to optimize delivery and deployment of their component's versions (when applicable).

The ARCADIAN-IoT consortium strives to adopt the **best practices and approaches** for the development of its framework and respective components. The twelve-factor *app*[22] (i.e.,

[22] https://12factor.net

approach) is one of such approaches. It is a methodology for building software-as-a-service solutions that, among others:

- Use declarative formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a clean contract with the underlying operating system, offering maximum portability between execution environments;
- Are suitable for deployment on modern cloud platforms, obviating the need for servers and systems administration;
- Minimize divergence between development and production, enabling continuous deployment for maximum agility;
- And can scale up without significant changes to tooling, architecture, or development practices.

The twelve-factor methodology can be applied to applications or services written in any programming language, and which use any combination of backing services (database, queue, memory cache, and others). Table 8 describes the methodology's 12 factors and depicts how ARCADIAN-IoT partners intend to adopt them.

Table 8 - Applicability of 12-Factor Approach in ARCADIAN-IoT

		Answers	**Likelihood**
I. Codebase - One codebase tracked in revision control, many deploys		14	70%
II. Dependencies - Explicitly declare and isolate dependencies		15	75%
III. Config - Store config in the environment		16	80%
IV. Backing services - Treat backing services as attached resources		5	25%
V. Build, release, run - Strictly separate build and run stages		11	55%
VI. Processes - Execute the app as one or more stateless processes		10	50%
VII. Port binding - Export services via port binding		10	50%
VIII. Concurrency - Scale out via the process mode		7	35%
IIX. Disposability - Maximize robustness with fast startup and graceful shutdown		8	40%
X. Dev/prod parity - Keep development, staging, and production as similar as possible		11	55%
XI. Logs - Treat logs as event streams		8	40%
XII. Admin processes - Run admin/management tasks as one-off processes		2	10%

To wrap-up, the answers collected via the survey have greatly contributed to extracting additional

insight and refining some architectural aspects. Details such as the types of communication mechanisms (i.e., message bus in combination with RESTful APIs), development preferences (e.g., programming languages, code repositories, open source or sub-component dependency), component storage requirements and others enable an informed decision making with respect to the inter-component communication depicted in the architecture.

The knowledge gathered from each of these aspects also helps paving the way to the start of the integration activities, as well as a well-informed component development - taking into account the overall architectural aspects. The following sub section will present the summary of the answers and main conclusions with respect to the deployment view.

5.2 Deployment View

The answers collected from the survey have also allowed the consortium to collect and assess deployment considerations such as target operating systems, type of software packaging (e.g., containers or binaries) and others. Like the previous section, the main questions and respective answers with respect to the deployment view are described next:

- **Is the horizontal scaling (e.g., replicating instances) of the component within the development roadmap in ARCADIAN?**

  Table 9 indicates that approximately half of Arcadian-IoT components will be able to horizontally scale when deployed. The numbers are expected as some components are deployed directly on the devices and naturally are not expected to scale horizontally. These conclusions are further validated by the answers provided by the next survey question (Table 10).

Table 9 - Horizontal Scaling

		Answers	Ratio
Yes		9	45%
No		11	55%

- **What is the deployment location of the component? If the component has different sub-components, with different deployment locations, please specify where each sub-component should be deployed.**
  As discussed in the previous point, approximately half of ARCADIAN-IoT will be fully deployed on the deployed on the devices or have at least one of its sub-components in the device or other sub-components elsewhere (e.g., Cloud). Table 10 shows the distribution of the deployment locations of ARCADIAN-IoT components, i.e. where each of the ARCADIAN-IoT components are expected to be supported.

Table 10 - Deployment Location

		Answers	Likelihood
IoT Device (e.g., sensor, drone, medical kit)		2	10%
IoT Gateway (e.g., smartphone)		4	20%
Network or Cloud		12	60%

Multiple		7	35%

Table 10 conveys that 2 of the components have their functionality exclusively running on the device. The answer to "Multiple" deployment locations indicates that 7 ARCADIAN-IoT components will have some of its sub-components (i.e., internal parts of each component) running on device as well as other locations (e.g., Hardened Encryption running on the device and on the Cloud). Additionally other components (or sub-components) may also be running just in the network (e.g., Network Flow Monitor) or in the Network and the device (e.g., eSIM). This information is further described and presented in Section 4.2 and by the end of this section.

- **What is the preferred operating system for deployment purposes (i.e., is what kind of OS is the component going to be deployed / which OS is it going to support)?** Regarding the target operating systems, Table 11 shows that most of ARCADIAN-IoT components target Linux-based operating systems (e.g., ubuntu). Additionally, partners also consider the need for development for mobile operating systems such as Android. This will be particularly relevant not only for ARCADIAN-IoT domains' demonstration but also for components such as the device behaviour monitoring which can also embed the IDS models within the mobile applications developed under each domain (e.g., Medical IoT).

Table 11 - Target Operating Systems

		Answers	**Ratio**
Linux		16	80%
RedHat		0	0%
Windows		0	0%
Other(s)		4	20%

- **Is the component suitable to be released as a service (if applicable and the deployment location allows it) and executed via an orchestration service?**

As Table 12 indicates, six ARCADIAN-IoT components are expected to be delivered as a service: Self-Recovery, Reputation System, Self-Aware Data Privacy, Biometrics, Hardened Encryption and Federated AI.

Naturally, most of the components do not have follow the same approach mainly due to deployment location (i.e., with some components being deployed on the devices and others in the network or cloud). This means that an orchestration service will be required as well as an appropriate packaging approach. Partners feedback on orchestration and packaging solutions is presented in the next two entries (Table 13 and Table 14).

Table 12 - Components deployable as a service

		Answers	**Ratio**
Yes		6	30%
No		14	70%

- **Where applicable, what kind of orchestration service is the most suitable?**

  Following on the previous question, ARCADIAN-IoT partners indicated that in the cases where orchestration is applicable, the most suitable solution would be Kubernetes. As Table 13 shows, there was also a preference on Docker Swarm and another answer indicating that is would be to be decided in the future. Nevertheless, the architecture of ARCADIAN-IoT Framework and its components is flexible enough to allow one or more orchestration services to support the service.

Table 13 - Orchestration preferences

		Answers	Preference
Kubernetes		4	20%
Docker Swarm		1	5%
OpenShift		0	0%
None		7	35%
Other		1	5%
No Answer		7	35%

- **How will the component be delivered (i.e., software packaging)?**

  Defining how the software is deliverable is an importance decision that has a long-term impact on integration and compatibility aspects. Table 14 shows that ARCADIAN-IoT partners opt to deliver software (i.e., ARCADIAN-IoT components) in the form of containers (e.g., docker containers), OS packages (e.g., .deb), binaries or other formats yet to be defined (via the GSMA-accredited RSP in the case of the installation of eSIM profile on the devices).

Table 14 - Packaging preferences

		Answers	Ratio
OS packaging (.deb, .rpm, .msi, etc)		4	20%
Container images (OCI, Docker, etc)		10	50%
Binaries		3	15%
Other		3	15%

- **How is the package of this component going to be distributed/stored?**

  The software solutions developed by ARCADIAN-IoT consortium are, at least during the project execution, privately distributed due to IPR matters. Table 15 shows that 7 components (i.e., Permissioned blockchain, Credentials Recovery, Verifiable Credentials, Attestation, IoT Device Behaviour Monitoring, Self-Aware Data Privacy and IoT Device Self-Protection) are expected to be made available in public repositories such as Docker Hub or similar. The remaining components will leverage private distribution repositories (e.g., JFrog Artifactory) and other approaches as a means of delivering their solutions to

authorized ARCADIAN-IoT actors.

Table 15 - Packaging storage and distribution

		Answers	Ratio
Public Repositories (e.g., Docker Hub)		7	35%
Private Repositories (e.g., JFrog Artifactory)		9	45%
Non-applicable		4	20%

At this stage of development and architectural definition, it was already possible to collect the minimum hardware requirements not only for the communication mechanisms (described in Section 4.1.2) but also for ARCADIAN-IoT components deployed in the cloud. Given the early stage of component development it was not possible to collect the minimum hardware requirements for all the components. Nevertheless, Table 16 depicts the expected minimum requirements for some of the components, namely, permissioned blockchain, verifiable credentials, self-recovery and reputations system. The requirements are modest and therefore do not imply significative efforts from the service providers. This information will be considered (and complemented) in later stages of the project, where the ARCADIAN-IoT framework is to be validated in Domain A, B and C.

Table 16 - Minimum Hardware Requirements for Cloud-based components

Component	CPU	RAM	DISK
Permissioned Blockchain	4 cores (e.g., AMD EPYC 7281)	8GB	100GB
Verifiable Credentials	4 cores (e.g., Raspberry Pi 4)	8GB	64GB
Self-Recovery	1 core (2-4 for heavier usage)	2GB (8GB for heavier use)	10GB
Reputation System	4 cores	0	500GB per domain

The survey concerning development and deployment views granted the consortium the ability to collect and reflect on the possible approaches, solutions, and additional component characteristics. As such, not only the architecture design reflected such considerations, as this process allowed these considerations to be fed towards the development and integration activities that will take place.

5.2.1 ARCADIAN-IoT component location

Figure 35 complements the architectural information provided in Section 2 (i.e., functional view and high-level deployment view). The figure illustrates the envisioned physical deployment location of each ARCADIAN-IoT component (without providing information at sub-component level). As it is possible to observe, part of ARCADIAN-IoT functionality will operate on the connectivity / network operator domain. This is the case of the Network Flow Monitor,

Authorization, Network Self-Protection and Network Self-Healing. These components contribute to the identity, security and recovery capabilities of the framework.
Moreover, IoT Devices are a simplified view of both sensors, IoT gateways, personal devices or drones addressed in the use cases. Future iterations (to be addressed in WP5) will be required in order to address the following points:

- include a per-domain representation of all ARCADIAN-IoT components and sub-components;
- specification of component for enabling interaction between ARCADIAN-IoT and external or inherited security functions (e.g. security proxies or gateway functions).

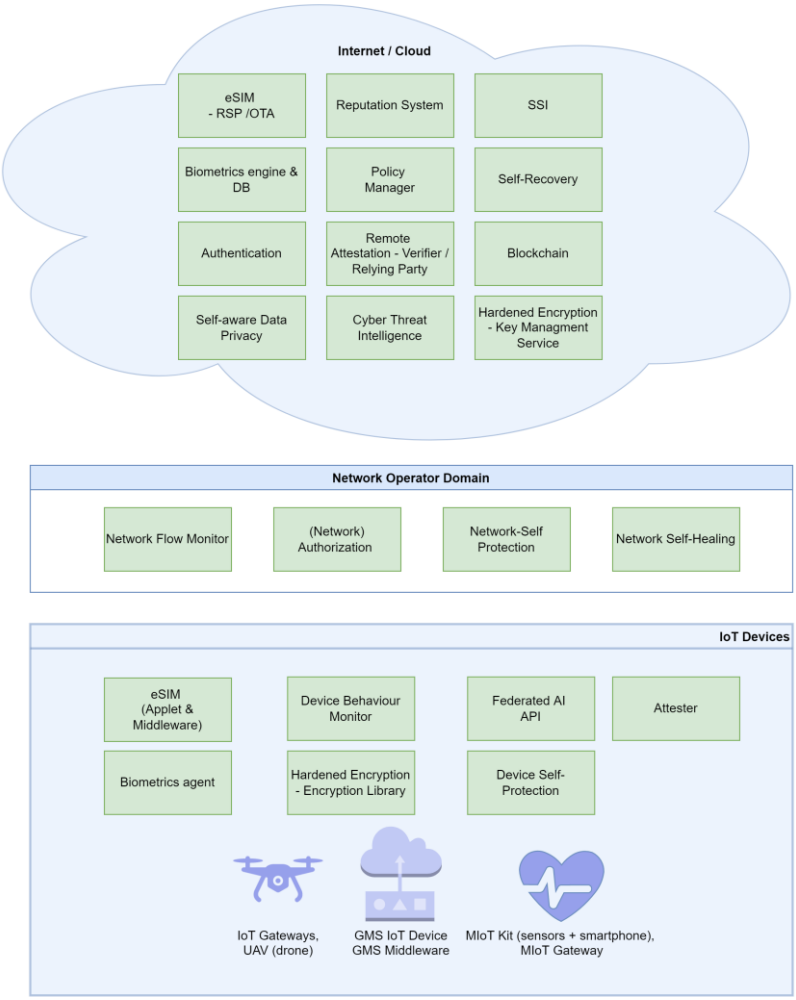

Figure 35 - ARCADIAN-IoT Deployment Locations

The following section addresses the operational and concurrency views.

6 ARCADIAN-IOT OPERATIONAL AND CONCURRENCY VIEWS

This section starts by presenting initial assumptions regarding ARCADIAN-IoT Framework operation. It addresses component concurrency and the level of dependency that components hold on each other (e.g., synchronization).

Taking into account ARCADIAN-IoT's nature and targeted outcomes – and associated TRL -, the main goal in doing so lies in presenting key identified / expected operational dependencies among its components. Such information is quite valuable for upcoming project stages, as it will feed both the integration planning in WP5 (e.g., the higher the dependency on a given component, the more crucial is the implementation, testing and validation of its functionality and interfaces for the integrated validation of other components) and the potential (joint or clustered) exploitation of results (in WP6). That being said, this section – and future updates to it - does NOT intend to comprehensively present all aspects required for ARCADIAN-IoT success in production environments.

6.1 Operational View

The operation and maintenance of ARCADIAN-IoT is not yet fully outlined at this stage of the project execution. One justification for this is that the project is still in a too early stage for enabling the definition of a system-wide strategy on how ARCADIAN-IoT will be controlled, managed and monitored. Additionally, the latter depends on clear exploitation approaches. As the consortium approaches the second third of the project, concrete exploitation strategies will need to be defined and stabilized - allowing the consortium to appropriately define not only the exploitation/commercialization strategies but also the way the framework should be operated and maintained, and by whom (if ARCADIAN-IoT project consortium will uncover a business champion among its partners).

Despite the current uncertainty, as the technical characteristics do not significantly limit the available options, it is currently possible to set out potential operation and maintenance strategies that the consortium may consider at later stages. Some of the possibilities can be described as follows:

- One of the options is for the administration and maintenance of the ARCADIAN-IoT Framework infrastructure to be directly provided by an entity (or multiple entities) offering the ARCADIAN-IoT solution. Such approach requires the necessary components and supporting services (e.g., message bus) to be deployed and visible within the ARCADIAN-IoT service.
- Alternatively, one may consider that different planes can be managed by distinct entities. For instance, it is reasonable to assume that the Identity Plane (representing identity provisioning and management services) could provide and managed by a single entity. In this case, the supporting services can be managed by any of the involved entities.
- The opposite approach or pathway is for each component to be provided and managed independently. This could be the case for specific ARCADIAN-IoT components such as CTI or Reputation System. Like the previous cases, the supporting services can be provided by any of the involved entities.

Other intermediate models can also be considered, such as exploiting the combination of information / alerting with decision entities (e.g., Behaviour Monitoring and IoT Device Self-protection), which would have obvious motivations (e.g., ensuring anomaly detection accuracy, and increasing trust in the service has direct benefits in the “quality” of a IoT device self-protection

entity). This pathway is being inherently implemented, in practice, in ARCADIAN-IoT project. Namely, the first activities in the integration roadmap for ARCADIAN-IoT framework will explore / address such groups of components with stronger dependencies on each other.

6.2 Concurrency View

The components of the ARCADIAN-IoT framework form a CoT that enables enhanced security, privacy and trust assurances to its users. As such, while some system components have isolated features and functionalities, others depend on other components in order to be fully operational. This is the case, for instance, of the blockchain and reputation system which are a primary source of information for components such as (network) Authorisation, Self-recovery or SSI (i.e., Verifiable credentials and Decentralized Identifiers).

Despite a certain degree of dependence, in the event of failures or inaccessibility of some components, the multi-component approach envisioned by ARCADIAN-IoT assures that most components can still operate and provide their intended functionalities (e.g., Behaviour Monitoring or Device Self-Protection). Additionally, the reliance on a common message bus for coordination and control (e.g., RabbitMQ of Kafka, as mentioned in section 4.1.2), is another feature that will allow several ARCADIAN-IoT components to still be able to function concurrently in the advent of anomalies.

7 CONCLUSIONS

This deliverable has presented the **ARCADIAN-IoT Framework architecture.** To seamlessly introduce the system architecture, Rozanski and Woods methodology was adopted – with the necessary adaptations to the R&I scope of the project. The document then lays out the motivation behind ARCADIAN-IoT and respective objectives, before dwelling in the architecture itself, and its **innovative components,** according to the plane their part of, and associated design and architectural specifications so far. ARCADIAN-IoT components are **structured by vertical and horizontal planes**, the former devoted to identity, trust and recovery management, which are complemented by the latter for managing privacy of data, security of components, as well as **decentralized storage** through blockchain technologies. Thus, ARCADIAN-IoT planes are presented not only to describe to objective of each plane and associated components, but also to lay the necessary background before introducing the system architecture with all ARCADIAN-IoT components. This background allows a natural presentation of the system architecture, which is then provided in two forms: (1) **functional view**, where the inter-component communication and **data flows** are presented; and (2) **deployment view**, where the **deployment location** of each component is represented.

This document also describes the **nature of the main data flows** (e.g., intrusion detection events, indicators of compromise or healing instructions) represented in ARCADIAN-IoT architecture. The system relies on message bus, REST APIs and DIDComm communications mechanisms to support **inter-component communication**. The main characteristics of such solutions are compared and the reasons for type of solutions is equally presented.

All **technical** partners (IPN, ATOS, LOAD, MAR, RGB, RISE, BOX2M, UWS, XLAB, TRU) were involved in the definition of ARCADIAN-IoT architecture, and E-LEX validated it from the **legal** point of view (particularly, but not exclusively addressing GDPR requirements). An iterative process with several rounds of interaction, allowed the consortium to reach a solid outcome in a substantially complex task given that ARCADIAN-IoT Framework contemplates **more than 20 components**.

The final sections of this document provide a first iteration of the **development** and **deployment** approach and preferences, aiming at preparing the overall development and integration of ARCADIAN-IoT Framework and its components. The operational and **concurrency** views, which considers the **operation** and **administration** of the framework, are equally addressed.

The next steps involve not only the development of ARCADIAN-IoT technical components, the initiation of the integration activities that will build the overall framework, and the preparation and implementation of the three domains where ARCADIAN-IoT framework will be validated: (A) Emergency and Vigilance; (B) Grid Infrastructure Monitoring; and (C) Medical IoT.

REFERENCES

[1] N. Rozanski and E. Woods, “Software systemas architecture: working with stakeholders using viewpoints and perpectives,” 2012.

[2] ARCADIAN-IoT, “D2.2 - Use case specification,” 2021.

[3] ARCADIAN-IoT, “D2.4 ARCADIAN-IoT framework requirements,” 2021.

[4] “W3C Decentralized Identifiers (DIDs) v1.0,” [Online]. Available: https://www.w3.org/TR/did-core/#did-controller. [Accessed 27 April 2022].

[5] “Sidetree Specification 1.0.0,” [Online]. Available: https://identity.foundation/sidetree/spec/. [Accessed 28 04 2022].

[6] “Decentralized Identifier Resolution (DID Resolution) v0.2,” [Online]. Available: https://w3c-ccg.github.io/did-resolution/. [Accessed 28 04 2022].

[7] T. L. O. T. Sam Curren, “DIF DIDCOMM Specification,” [Online]. Available: https://identity.foundation/didcomm-messaging/spec/.

[8] “W3C, Verifiable Credentials Data Model v1.0,” [Online]. Available: https://www.w3.org/TR/vc-data-model/. [Accessed 21 11 2021].

[9] K. J., X. Z., N. D., X. S. and Z. J., “Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory,” *IEEE Internet of Things,* vol. 6, no. 6, pp. 10700-10714, 2019.

[10] A. Josand and R. Ismail, “The beta reputation system,” in *15th Bled Electron. Commer. Conf.*, 2002.