



Grant Agreement N°: 101020259
Topic: SU-DS02-2020



ARCADIAN-IoT

Autonomous Trust, Security and Privacy
Management Framework for IoT

D4.3: ARCADIAN-IoT Vertical Planes – Final version

Revision: v1.3

Work package	4
Task	Tasks 4.1, 4.2, 4.3, 4.4, and 4.5
Due date	31/01/2024
Submission date	31/01/2024
Deliverable lead	XLAB
Version	1.3
Partner(s) / Author(s)	ATOS: Ross Little IPN: Sérgio Figueiredo, Rúben Leal 1GLOBAL: João Casal, Afonso Paredes, Ivo Vilas Boas, Jailson Cavalcanti UWS: Jose M. Alcaraz Calero, Qi Wang, Ignacio Martinez-Alpiste, Julio Diez Tomillo, Rafael Fayos Jordan XLAB: Nejc Bat, Jan Antic UC: Bruno Sousa, Luis Paquete, João Nunes

Abstract

This public technical report constitutes deliverable D4.3 of ARCADIAN-IoT, a Horizon2020 project with the **grant agreement number 101020259**, under the topic **SU-DS02-2020**. D4.3 has the purpose of reporting the research activities performed with respect to the development of the ARCADIAN-IoT vertical planes (Identity Management, Trust Management and Recovery Management) and the associated components.

Keywords: ARCADIAN-IoT, Identity, Trust, Recovery, Management

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.8	20/12/2023	Inputs by all partners inserted, ready for final consolidation	XLAB. all
V0.9	12/01/2024	Consolidated and integrated version for internal review	XLAB, all
V1.1	24/01/2024	Internal Review done	IPN
V1.2	30/01/2024	Revised version after internal review	XLAB, all
V1.3	31/01/2024	Final version with last check and revision implemented, ready for submission	XLAB

Disclaimer

The information, documentation and figures available in this deliverable, is written by the ARCADIAN-IoT (Autonomous Trust, Security and Privacy Management Framework for IoT) – project consortium under EC grant agreement 101020259 and does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.

Copyright notice: © 2021 - 2024 ARCADIAN-IoT Consortium

Project co-funded by the European Commission under SU-DS02-2020

Nature of the deliverable: OTHER

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 D4.3 is the final deliverable reporting the up-to-date technical research activities performed regarding the Vertical Planes of ARCADIAN-IoT. It thus comprises the analysis and investigations on how to provide each of the components for the three Vertical Planes – Identity Management, Trust Management, and Recovery Plane - taking into consideration the defined framework requirements and applicable domains and specific use cases.

This deliverable is an update to the previous Deliverable 4.2, which presented the interim results focused on the 1st prototype (P1). D4.3 stands as self-contained document, providing a comprehensive report on the development of each WP4 Vertical Plane component, without the need to refer to previous deliverables.

The three vertical planes contribute towards the same objective, with some components being use case agnostic and others being more applicable to certain use cases. The associated components are as follows:

- Identity Management Plane (Task 4.1)
 - Decentralized Identifiers
 - eSIM – hardware-based identity and authentication
 - Biometrics
 - Authentication
- Trust Management Plane (Task 4.2)
 - Verifiable Credentials
 - Network-based Authorization
 - Reputation System
 - Remote Attestation
- Recovery Management Plane (Task 4.3)
 - Self-recovery
 - Credential recovery

The main outcome of this deliverable includes the updated specification of each of the components and includes architecture design, interfaces and APIs, relevant security considerations and status towards the support of the targeted functionalities. Part of the components also provide additional details such as the description of implementation approach (e.g. target technology or software language), adopted approach for supporting the project's use cases in the three addressed domains, and pointers to resulting resources. Another major outcome includes preliminary or partial evaluation results for components which are in more advanced implementation state.

Finally, the report considers the future work that is to be taken towards the completion for each component (e.g. currently missing functionalities, interfaces or supported execution environments) and which will enable their future integration, validation and evaluation.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	4
TABLE OF CONTENTS.....	5
LIST OF FIGURES.....	8
LIST OF TABLES	11
ABBREVIATIONS.....	12
1 INTRODUCTION	14
1.1 ARCADIAN-IoT and its Vertical Planes	14
1.2 Objectives.....	15
1.3 Component description methodology.....	16
2 IDENTITY PLANE.....	17
2.1 Decentralized Identifiers.....	17
2.1.1 Overview	17
2.1.2 Technology research	18
2.1.3 Design specification	27
2.1.4 Evaluation and results	37
2.1.5 Future work.....	37
2.2 eSIM – Hardware-based identification and authentication	38
2.2.1 Overview.....	38
2.2.2 Technology research	41
2.2.3 Design specification	45
2.2.4 Evaluation and results	46
2.2.5 Future work.....	47
2.3 Biometrics	47
2.3.1 Overview.....	47
2.3.2 Technology research	50
2.3.3 Design specification	54
2.3.4 Evaluation and results	59
2.3.5 Future work.....	59
2.4 Authentication	60
2.4.1 Overview.....	60
2.4.2 Technology research	61
2.4.3 Design specification	63
2.4.4 Evaluation and results	67
2.4.5 Future work.....	68
3 TRUST PLANE.....	69
3.1 Verifiable Credentials.....	69
3.1.1 Overview	69
3.1.2 Technology research	70
3.1.3 Design specification	75
3.1.4 Evaluation and results	98
3.1.5 Future work.....	98
3.2 Authorization: Network-based Authorization enforcement and authorization distribution.....	100
3.2.1 Overview.....	100
3.2.2 Technology research	101

3.2.3	Design specification	106
3.2.4	Evaluation and results	109
3.2.5	Future work.....	109
3.3	Reputation System	110
3.3.1	Overview	110
3.3.2	Technology research	111
3.3.3	Design specification	119
3.3.4	Evaluation and results	123
3.3.5	Future work.....	125
3.4	Remote Attestation.....	126
3.4.1	Overview\.....	126
3.4.2	Technology research	128
3.4.3	Design specification	133
3.4.4	Evaluation and results	140
3.4.5	Future work.....	140
4	RECOVERY PLANE.....	141
4.1	Self-recovery	141
4.1.1	Overview	141
4.1.2	Technology research	142
4.1.3	Design specification	143
4.1.4	Evaluation and results	145
4.2	Credentials recovery	145
4.2.1	Overview	145
4.2.2	Technology research	146
4.2.3	Design specification	148
4.2.4	Evaluation and results	152
4.2.5	Future work.....	152
5	CONCLUSIONS	153
	APPENDIXES	154
	Appendix A – Analysis of events in Domain A for Reputation System	154
	Appendix B – Analysis of events in Domain B for Reputation System	155
	Appendix C – Analysis of events in Domain C for Reputation System.....	156
	Appendix D – Policy Manager User Manual	157
	Policy Management Dashboard	157
	Default policy	157
	Updating policies	157
	Deleting policies\.....	158
	Adding new policies.....	158
	Policy API.....	158
	Policy fields.....	158
	Backend fields	158
	Policy definition fields.....	158
	Endpoints	159
	GET /policies	159
	POST /policy.....	159
	PATCH /policy/{id}	159
	DELETE /policy/{id}	159
	GET /default_policy	159
	PUT /default_policy	159

Appendix E – DID METHODS.....	160
REFERENCES.....	163

LIST OF FIGURES

Figure 1 – ARCADIAN-IoT framework	14
Figure 2 - ARCADIAN-IoT DID solution's basic architecture model [3]	17
Figure 3 - Sidetree DID Method Overlay network [16]	22
Figure 4 - Public DIDs published on Sidetree Node based on Transmute Sidetree.js [38]	28
Figure 5 - Public DIDs published on ARCADIAN-IoT Permissioned Blockchain	29
Figure 6 - Public DID:WEB hosted on Service Provider's endpoint to provide service authentication	30
Figure 7 - Privacy preserving peer DID created in SSI wallet per connection	32
Figure 8 – Public DID published on the Permissioned Blockchain	32
Figure 9 – Public did:priv method example	33
Figure 10 – DID Challenge / Response (Ref: [48])	34
Figure 11 Issued Verifiable Credential with BBS+ proof	35
Figure 12 Presented derived Verifiable Credential with BBS+ proof	36
Figure 13 - eSIM component overall view	39
Figure 14 - Network-based authentication component baseline	43
Figure 15 - Architecture of the Network-based authentication in third-party services	44
Figure 16 - Logical process view of Biometrics component.	55
Figure 17 - Sequence diagram for registration use case.	56
Figure 18 - Sequence diagram for person authentication from a smartphone.	57
Figure 19 - Sequence diagram for person authentication from a video being recorded by drone.	57
Figure 20 - High-level architectural view of the biometrics component and other components that have relation with.	58
Figure 21 – AMQP API specification for biometrics messaging	59
Figure 22 – REST API specification for Biometrics component	59
Figure 23 - ARCADIAN-IoT authentication high-level architecture	63
Figure 24 - Architecture from ARCADIAN-IoT MFA for persons	65
Figure 25 - Ledger uSelf built on top of Hyperledger Aries GO Agent	71
Figure 26 - Protocol support for SSI [32]	73
Figure 27 - Cryptographic technology [32]	73
Figure 28 - Register Person in ARCADIAN-IoT Framework by a SP service	76

Figure 29 - Ledger uSelf Broker Self-Sovereign Identity Solution + SSI IdP	77
Figure 30 - Issue a Person VC	79
Figure 31 Issue an Organization Member VC	80
Figure 32 Figure 30 - Issue a Device VC	81
Figure 33 – Present and verify a Person VC	82
Figure 34 – Present and verify an Organization Members VC	83
Figure 35 Present and verify a Device VC	84
Figure 36 Authenticate a Constrained IoT Device with its Public DID	85
Figure 37 - Service Provider service registers a Person	86
Figure 38 - Service Provider service registers an Organization Member	87
Figure 39 - Service Provider registers an IoT Device	88
Figure 40 - Service Provider deletes a registered Person that it previously registered	89
Figure 41 - SSI IdP Interface description	90
Figure 42 - Self-Sovereign Identity deployment in the ARCADIAN-IoT Framework	91
Figure 43 - SSI IdP Registered Entities	92
Figure 44 - SSI IdP Issuer Screen	93
Figure 45 - QR code display to connect to the SSI Agent	93
Figure 46 - Ledger uSelf mobile SSI Wallet UI	94
Figure 47 - SSI IdP Data Model	97
Figure 48 - 3GPP's PCC Architecture overview ⁵²	103
Figure 49 - Open5GS architecture	104
Figure 50 - ARCADIAN-IoT Network-based Authorization high-level architecture	107
Figure 51 - Network-based Authorization technical architecture	108
Figure 52 - Flow of processing events in the reputation system	113
Figure 53 - Mechanisms in Blockchain for GDPR compliance	116
Figure 54 - Developed architecture to allow the compliance with GDPR (PDA – Personal Data Analyzer, JCA – Joint Contract Agreement)	117
Figure 55 - Screen of reputation system	118
Figure 56 - Parallel coordinates chart with brushed events	118
Figure 57 - Reputation System & Policy Manager logical architecture view	120
Figure 58 - Reputation System internal logic to determine reputation score	121
Figure 59 - Comparison of reputation score with different severity factor	124
Figure 60 - OWASP's top 10 IoT security issues (2018 version)	128
Figure 61 - Remote Attestation logical architecture and external dependencies	134

Figure 62 - Verifier's Internal logical architecture	134
Figure 63 - Remote attestation bootstrap & execution procedure signaling	135
Figure 64 - Remote Attestation for ARCADIAN-IoT (RA2IoT) deployment architecture view	138
Figure 65 - Self-recovery logical architecture view	143
Figure 66 - Simple recovery case	144
Figure 67 - Auto-triggered recovery	144
Figure 68 - Recovery of SSI Wallet Credentials Use Case	149
Figure 69 Recovery of IoT Device Credentials Use Case	150
Figure 70 - SSI Credential Recovery Logical Architecture	150
Figure 71 IoT Device Credential Recovery	151
Figure 72 - Screen with default and other policies	157
Figure 73 – Configurable policies fields	158

LIST OF TABLES

Table 1 - Network-based authentication interfaces	46
Table 2: Image databases	51
Table 3 – Face verification algorithm accuracy	51
Table 4 - Face detection algorithm accuracy	52
Table 5: Machine Learning frameworks	52
Table 6 - MFA interfaces	64
Table 7 - Network-based Authorization interface to external components	107
Table 8 - Data privacy concerns (preliminar analysis)	115
Table 9 - Technologies in the reputation system and policy manager	121
Table 10 - Technologies in the reputation system and policy manager	122
Table 11 - Results of Alpha and Beta testing (with forgetting factor of 0.5)	123
Table 12 - Comparison between models with (a) negative events, and (b) with 20 random events	124
Table 13 DID Method Table [8]	160

ABBREVIATIONS

5GC	5 th Generation Core
AI	Artificial Intelligence
AIoT	ARCADIAN-IoT
aiotID	ARCADIAN-IoT Identifier
CAS	Content Addressable Storage
CBOR	Concise Binary Object Representation
CTI	Cyber Threat Intelligence
CWT	CBOR Web Token
DAG	Directed Acyclic Graph
DB	Database
DID	Decentralized Identifier
DID Doc	DID Document
DGA	Drone Guardian Angel
DPoP	Demonstrating Proof-of-Possession at the Application Layer
DTR	Device Trust Registry
ECDSA	Elliptic Curve Digital Signature Algorithm
eSIM	embedded Subscriber Identity Module
eUICC	embedded Universal Integrated Circuit Card
EPC	Evolved Packet Core
FE	Functional Encryption
GSMA	Global System for Mobile Communications Association
GSMA-SAS	GSMA's Security Accreditation Scheme
GPU	General Processor Unit
GUI	Graphical User Interface
HD	High Definition
HE	Hardened Encryption
HTTP	Hypertext Transfer Protocol
HW	Hardware
IMSI	International Mobile Subscriber Identity
IdP	Identity Provider
IDS	Intrusion Detection System
IoT	Internet of Things
IOTA	Internet of Things Association
IPR	Intellectual Property Rights
IPFS	Inter Planetary File System
JSON	JavaScript Object Notation
JWM	JSON Web Message
JWT	Java Web Token
KPI	Key Performance Indicator
LTE	Long-Term Evolution
NIST	National Institute of Standards and Technology

NR	New Radio
OCS	Online Charging System
OFCS	Offline Charging System
OIDC	OpenID Connect
OS	Operating System
OSD	Object Storage Daemon
PCC	Policy and Charging Control
PCF	Policy Control Function
PCEF	Policy and Charging Enforcement Function
PCRF	Policy and Charging Rules Function
RA	Remote Attestation
RA2IoT	Remote Attestation for ARCADIAN-IoT
RATS	Remote Attestation Procedures
REST	Representational State Transfer
RFC	Request for Comment
RoT	Root of Trust
SAS	Security Accreditation Scheme
SE	Secure Element
SIOP	Self-issued OpenID Provider
SOTA	State-Of-The-Art
SP	Service Provider
SSI	Self-Sovereign Identity
UCCS	Unprotected CWT Claims
VC	Verifiable Credential
VDR	Verifiable Data Registry
XML	eXtensible Markup Language

1 INTRODUCTION

1.1 ARCADIAN-IoT and its Vertical Planes

The ARCADIAN-IoT project aims to develop a cyber security framework relying on a novel approach to manage and coordinate, in an integrated way, identity, trust, privacy, security, and recovery in IoT systems. The proposed approach organizes the multiple cyber security functionalities offered by the framework into several planes combined in an optimized way to support the end-to-end services. In particular, the framework includes three Vertical Planes devoted to identity, trust, and recovery management, and three Horizontal Planes supporting the Vertical Planes by managing privacy of data, monitoring security of entities, and providing Permissioned Blockchain and Hardened Encryption technologies (see Figure 31).

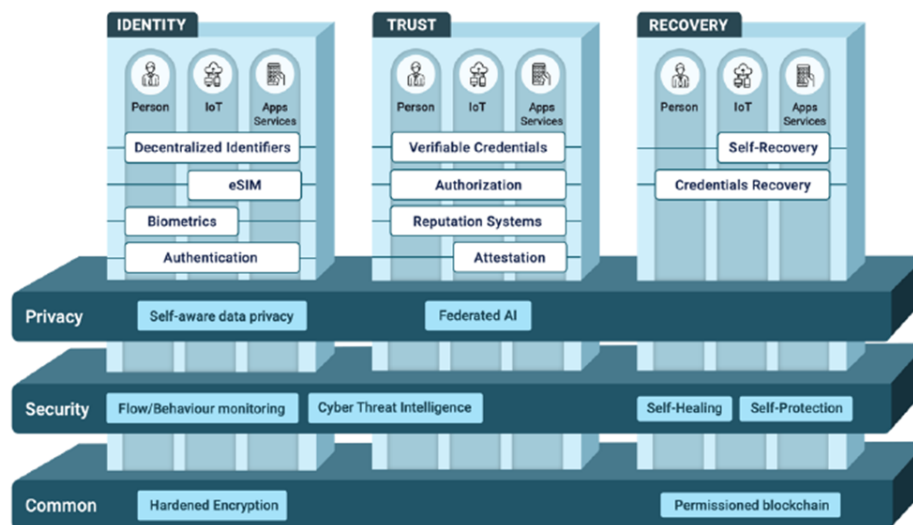


Figure 1 – ARCADIAN-IoT framework

Work Package 4 (WP4) in the ARCADIAN-IoT project is dedicated to the design and technological development of the functionalities mapped into the Vertical Planes for each selected use case. It is organized in three tasks, each one focusing on one plane. The research activity in WP4 is conducted from October 2021 to January 2024, and this deliverable (D4.3) details the research activities, the provided resources, and the results of evaluations obtained within WP4 until January 2024. The reporting takes an agile approach, with D4.3 fully revising D4.2 [46] to provide a self-contained report on the components in the Vertical Planes and including greater detail on their design and implementation. The project has setup a Gitlab repository¹ where partners - where there is no IPR restriction - have shared pertinent resources (e.g. component's OpenAPI interface specification, code snippets or in some cases component software).

The Vertical Planes of the ARCADIAN-IoT framework are organized as follows:

- The **Identity plane** enables the management of identities of the different entities (e.g. persons, devices and ARCADIAN-IoT components), and comprises work on the multiple identification schemes, particularly the **Decentralized Identifiers** for providing a decentralized digital identity, **eSIMs** as secure elements capable of storing identity and authentication credentials, and **Biometrics** focusing facial recognition from different devices and considering diverse circumstances (e.g. distance, angle, exposure to light). The status of the Identity Plane is presented in section 2.
- The **Trust plane** implements mechanisms for managing trust on the involved entities (persons, devices and services), namely **Verifiable Credentials** as a method to enable

¹ https://gitlab.com/arcadian_iot/

trusted identification of users and things through the issuing of identity claims, **Remote Attestation** for attesting IoT devices and services integrity with the support of hardware-based RoT, **Network-based Authorization** for enforcing trust-based authorization rules in the network core and informing secure elements about their corresponding device's trustworthiness level, and the **Reputation System**, responsible for determining the different entities' Reputation scores based on data received from other entities and ARCADIAN. The research status on Trust plane is described in section 3.

- Finally, the **Recovery plane** addresses recovery management of data associated to the different types of entities, concretely the **Self-Recovery** for enabling heterogeneous devices to access data recovery services according to different access policies, and the **Credentials Recovery** for secure recovery of credentials, the first and necessary step to trigger data recovery actions. The research status on Recovery plane is presented in section 4.

1.2 Objectives

WP4 aims at contributing to achieving 6 of the ARCADIAN-IoT's objectives and associated individual Key Performance Indicators (KPIs), as defined in its grant agreement. Furthermore, the component-specific KPIs have been revised in this deliverable for providing more accurate and measurable indicators for the success of the project.

The summary of the project's objectives and WP4's contribution to the associated KPIs is listed below, with additional details being given under each WP4 component's section:

- **Objective 1: To create a decentralized framework for IoT systems - ARCADIAN-IoT framework**
- **Objective 2: Enable security and trust in the management of objects' identification**
 - To support at least 2 identification factors for devices
 - To support Decentralized Identifiers in at least two of the use case domains
 - To use eSIM to support an identity approach at hardware level, as a robust identity mechanism for devices
- **Objective 3: Enable distributed security and trust in management of persons' identification**
 - To enable at least 3 multiple simultaneous identification approaches for persons (Decentralized Identifiers, SIM of personal device and Biometrics)
 - To reduce inference time for face verification algorithms, reduce end-to-end speed of biometrics process and improve accuracy and reliability of face verification algorithms at close and far distances
 - To enable cost-effective camera and drone platforms for person identification purposes
- **Objective 4: Provide distributed and autonomous models for trust, security and privacy – enablers of a Chain of Trust (CoT)²**
 - To support Verifiable Credentials (VC) protocol for integration in the Permissioned Blockchain in at least one domain use case, and supporting VC's interoperability with at least one eIDAS identity schema
 - To support automatic bidirectional communication authorization enforcement for devices and people according to trustworthiness levels and its dynamic changes related with security events, and reduce authorization policies enforcement time after the network is informed

² A detailed analysis of how ARCADIAN-IoT enables the different CoTs for IoT is provided in deliverable D5.2

- To be able to determine and share reputation score for persons, devices and services
- To increase the supported number of reputation events received and processed per unit of time and to reduce the time required to determine reputation
- To support Remote and functional attestation providing Root of Trust mechanisms with at least one type of Secure Element (eSIM, cryptochip) and for at least two different types of IoT devices
- To support remote attestation of devices through multiple verifiers (by leveraging Attribute-based encryption for attestation evidence)
- To support initiation of remote attestation via watchdog-based attestation trigger (via Verifier) and attestation cues (via Reputation System)
- To influence device and service reputation models via Remote Attestation
- **Objective 5: Provide a Hardened Encryption with recovery ability.**
 - To enable data to be encrypted in a selective way, by applying policies that define which stakeholders can decrypt partial or complete data
- **Objective 6: Self and coordinated healing with reduced human intervention.**
 - To enable applications/processes/devices to run as expected after recovery
 - To support Credential recovery operations after security / privacy incidents with persons and IoT devices, and to support recovery of DIDs and VCs
 - To securely inform the eSIM of devices trustworthiness level
 - To use eSIM in device self-protection and self-recovery actions

1.3 Component description methodology

The work associated to each ARCADIAN-IoT component is described in the upcoming sections according to the following structure:

- First, the **Overview** provides the overall research and development scope associated to the component, including requirements, innovation-driven objectives and associated KPIs.
- The **Technology Research** spans both background relevant to the work performed in ARCADIAN-IoT (e.g. prior related solutions or knowledge), key research findings and achievements attained during the project, as well as resources produced as outcomes from the research and development work (e.g. datasets or software code).
- **Design specification** describes the component from a software design perspective, presenting relevant specifications such as logical or deployment architectures, internal and external interfaces, or other technical specifications specific to the scope of the research area.
- In **Evaluation and results**, performed experiments and associated evaluation results are described.
- Finally, in **Future Work**, the research and improvement opportunities identified as a result of ARCADIAN-IoT work are presented.

2 IDENTITY PLANE

2.1 Decentralized Identifiers

2.1.1 Overview

2.1.1.1 Description

As described in the W3C DID Core Specification [3] *“Decentralized identifiers (DIDs) are 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 resolves to DID Documents. The DID solution will follow the standard architecture model as portrayed by the following figure in the DID Core specification:

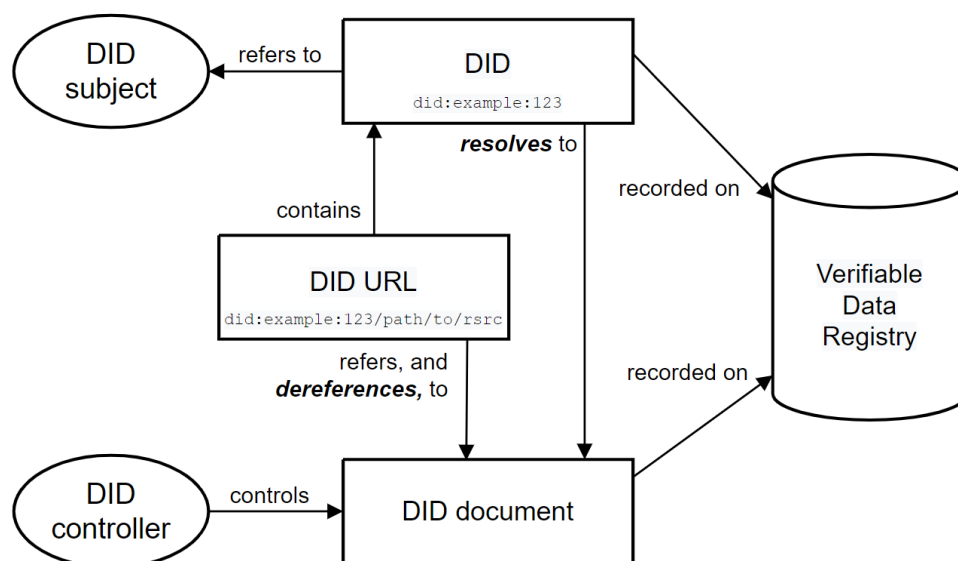


Figure 2 - ARCADIAN-IoT DID solution's basic architecture model [3]

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. The primary candidate VDRs analysed were based on sidetree DID Method overlay networks composed of independent peer nodes, with their trust anchor provided by blockchain. These nodes implement Content Addressable Storage (CAS) to host the DID Docs and interact with a blockchain to provide a notarised trust anchor, as described in the Sidetree specification. That said, other DID methods were also analysed for their suitability to the ARCADIAN-IoT framework and use cases, considering that latest DID methods that are trusted, provide privacy and do not rely upon blockchain.

It was proposed therefore to provide ARCADIAN-IoT framework with different options for supporting DIDs as per the needs of use case deployments. Specifically, DIDs are part of the Self-Sovereign Identity solution ARCADIAN-IoT framework, as they provide the root of trust in Verifiable Credentials as described in section 3.1.

2.1.1.2 Requirements

A recall of the high-level requirement 1.1.1 first defined in D2.4 is included below and it is also supplemented with additional related sub-requirements.

- Requirement 1.1.1 – Decentralized Identity Management
 - o Decentralised Identifiers (DID) to be supported as per the W3C Decentralized Identifier specification.
 - o Support cryptographic mechanisms such as zero knowledge proof (ZKP) and ZK-SNARKS that add advanced privacy capabilities
 - o Make use of DLT blockchain technologies in providing Decentralized Identifiers.
 - o Connection with an existing distributed and decentralised node for storing the ledger information on which the Self-Sovereign Identity) SSI system will rely.
 - o Creation of a mobile interface for the end user's personal devices.

2.1.1.3 Objectives and KPIs

KPI scope	
To support at least two of the use case domains	
Measurable Indicator	
Number of domains using DIDs	
Target value	Achieved value
2	3

KPI scope	
Enable, at least 3 multiple simultaneous identification approaches for persons.	
Measurable Indicator	
Implement Decentralized Identifiers in a person's mobile wallet as a basis for supporting Verifiable Credential identification	
Target value	Achieved value
3	3

KPI scope	
Support, at least two robust identity mechanisms for devices and apps/services.	
Measurable Indicator	
Devices and apps/services support Decentralized Identifiers.	
Target value	Achieved value
2	2

2.1.2 Technology research

Candidate Verifiable Data Registries (VDRs) that was under analysis includes distributed Sidetree DID Method overlay networks composed of independent peer nodes, with their trust anchor provided by blockchain. These nodes implement CAS to host the DID Docs and interact with a blockchain to provide a notarised trust anchor, as described in the Sidetree specification [4]. That said, other DID methods are also analysed for their suitability to the ARCADIAN-IoT framework, and also for supporting different DID methods as per the needs of differing scenarios.

To this effect, ARCADIAN-IoT will consider supporting Decentralized Identifiers by the following methods and analyse the pros and cons of each:

- I. Integrating with an existing distributed and decentralised system for storing the DID Doc on which the SSI system will rely (external to ARCADIAN-IoT components)
- II. Integrating with the Permissioned Blockchain (developed in WP3) to provide a trust anchor for publishing the DID Doc
- III. Integrating with self-published DIDs that do not rely upon existing distributed and decentralised systems

2.1.2.1 Background

ATOS have previous integration knowledge and developed java code supporting HTTP Signatures [39] for client and server side implementations with public keys published by DID:WEB [10]. Additionally, ATOS have an existing SSI Wallet prototype as part of the labs Ledger uSelf solution based on Hyperledger Aries (see section 3.1.2.1 for more details).

The following sections investigate the current state of the art in this area.

2.1.2.1.1 Integration with existing distributed and decentralised systems supporting DIDs

As can be seen from Table 13 DID Method Table in Appendix E (obtained from W3C DID Specification Registries [8]), there are over a hundred published DID methods utilising different VDR technologies to host the DIDs and others that don't need any VDR. Previously it was thought to publish public DID Docs directly on a blockchain network such as with did:sov or did:signor. However, ARCADIAN-IoT will not follow this approach so to avoid any potential issue with the GDPR and also to consider more recent advancements in this area to host the DID docs off-chain, but still provide the necessary trust anchor by different means e.g., decentralised, distributed.

We will examine some of these DID methods that could be well suited to the needs of ARCADIAN-IoT as follows:

did:elem [9]

The DID method element is an implementation based on the Sidetree protocol that uses the public Ethereum blockchain as the ledger layer and IPFS as a Content-addressable storage layer. Tools are made available for users to manage their own DIDs.

The primary benefit of using this method is that the DID Doc's are hosted off-chain with their trust anchored in the Ethereum blockchain network, and thus personal DID Doc data can be deleted. It is also possible to install the software to setup a private network and integrate this into an ARCADIAN-IoT deployment, as described in section 2.1.2.2.

It could be thought that a potential disadvantage is that, as it is hosted on a public blockchain, then potentially if a hacker managed to gain access to the user's DID Doc, he could update the keys to use the ones he has control of. However, as the ability to modify the DID Doc is based upon the user having access to the private key for controlling the DID, the risk is actually the same whether it is a permissioned blockchain or a public blockchain. Note, the recovery procedure would also be the same in that a DID controller (third person) would use a recovery key to regain control of the DID Doc if this scenario were to occur.

did:web [10]

DIDs that target a distributed ledger face significant practical challenges in bootstrapping enough meaningful trusted data around identities to incentivize mass adoption. This DID method simply bootstraps the trust using a web domain's existing and well-known address to host and manage the DIDs, as per the following examples.

Example of an organisation decentralized identifier:

- did:web:w3c-ccg.github.io

Example of an organisation member decentralized identifier:

- did:web:w3c-ccg.github.io:user:alice

This is a very simple method where the above example organisation DID Doc would be hosted at <https://w3c-ccg.github.io/well-known/did.json>. This would enable organisations to easily manage their own DIDs for persons, things and services and only the organisation's themselves can update the DID Doc.

did:ion [11]

ION is a Layer 2 open, permissionless network based on the purely deterministic Sidetree protocol, which requires no special tokens, trusted validators, or additional consensus mechanisms; the linear progression of Bitcoin's timechain is all that is required for its operation. ION is a public, permissionless, DID network developed by Microsoft that implements the blockchain-agnostic Sidetree protocol on top of Bitcoin (as a 'Layer 2' overlay) to support DIDs/DPKI (Decentralized Public Key Infrastructure) at scale, where the DID Docs are hosted off-chain on the IPFS.

The majority of ION's code is developed under the blockchain-agnostic Sidetree protocol repository³, which the project uses internally with the code required to run the protocol on Bitcoin, like the ION network.

It is therefore similar to the did:elem method previously described and is able to be supported by integrating to a node hosted by Microsoft or alternatively installing a bitcoin node and Sidetree deployment. The tools support for creating and publishing DIDs with ION are available online⁴, and it is seen that the native key algorithms supported are: secp256k1 and Ed25519.

did:ebis [12][14]

European Union is supporting the adoption of Self-Sovereign identity under the European Blockchain Services Infrastructure⁵ (EBSI) and within that initiative the European Self-Sovereign Identity Framework⁶ (ESSIF). EBSI provides a blockchain infrastructure that offers cross-border public services based on Hyperledger Besu. DIDs are created with the Besu blockchain addresses and hosted on the blockchain itself. The use of DIDs is aimed at trusted services and for natural and legal person identifiers.

Currently services are under development and are restricted to a selected group of projects as early adopters [19] and organisations that want to test their wallets with the ecosystem.

Within ARCADIAN-IoT a sub-objective is to support eIDAS Bridge [20] within the ESSIF project where a service can issue Verifiable Credentials to a user.

An important note on the integration to support the did:ebis is the need to perform EBSI DID authentication^[9] with the SIOP protocol as opposed to DIDCOMM so it would be needed for the SSI wallet to be compliant with the former. Also, of note is that the cryptographic key algorithm supported by EBSI at this time is secp256k1.

As the open source SSI frameworks under consideration to support Verifiable Credentials in section 3.1. only supports DIDCOMM, at this time, it will be a challenge to support integration with EBSI considering also it would be needed to apply to be an early adopter. A future action will be to consider how SIOP can be integrated as an additional DID messaging protocol in the SSI Frameworks under consideration.

did:iota [13]

The IOTA DID Method Specification describes a method of implementing the Decentralized Identifiers standard on the IOTA Tangle, a Distributed Ledger Technology discussed in ARCADIAN-IoT D3.1 [5]. It currently conforms to an outdated version of the W3C DID

³ <https://github.com/decentralized-identity/Sidetree>

⁴ <https://github.com/decentralized-identity/ion-tools#ionjs>

⁵ <https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Home>

⁶ <https://decentralized-id.com/government/europe/eSSIF>

specifications v1.0 Working Draft 20200731 and describes how to publish DID Document Create, Read, Update and Delete (CRUD) operations to the IOTA Tangle. In addition, it lists additional non-standardized features that are built for the IOTA Identity implementation.

Important features of IOTA Tangles are:

- The lack of fees, requiring no cryptocurrency tokens to be owned in order to submit a message to the DLT.
- The DLT supports both a public and permissionless network which runs the IOTA cryptocurrency.

The DIDs that follow this method have the following format:

```

iota-did = "did:iota:" iota-specific-idstring
iota-specific-idstring = [ iota-network ":" ] iota-tag
iota-network = char{,6}
iota-tag = base-char{44}
char = 0-9 a-z
base-char = 1-9 A-H J-N P-Z a-k m-z
  
```

iota-network

This is an identifier of the public or private (permissionless or permissioned) IOTA network where the DID is stored.

The following values are reserved:

- main: This references the main network which refers to the Tangle known to host the IOTA cryptocurrency.
- dev: This references the development network known as "devnet" maintained by the IOTA Foundation.

When no IOTA network is specified, it is assumed that the DID is located on the main network. This means that the following DIDs will resolve to the same DID Document as in the following example:

Example:

- did:iota:main:H3C2AVvLMv6gmMnam3uVAjZpfkcJCwDwnZn6z3wXmqPV
- did:iota:H3C2AVvLMv6gmMnam3uVAjZpfkcJCwDwnZn6z3wXmqPV

IOTA-Tag

The IOTA tag references an indexation which resolves to the initial DID Messages, and the following steps MUST be taken to generate a valid tag:

- Generate an asymmetric keypair using a supported verification method type.
- Hash the public key using BLAKE2b-256 then encode it using Base58-BTC.
- This public key MUST be embedded into the DID Document (see CRUD: Create).

DID Documents associated with the did:iota method consist of a chain of data messages, called "DID messages", published to a Tangle. The Tangle has no understanding of DID messages and acts purely as an immutable database. The chain of DID messages and the resulting DID Document must therefore be validated on the client side. Therefore any agent that needs to read and verify a did:iota method will need to implement specific Tangle validation.

The IOTA Identity framework currently supports two Verification Method Types:

- Ed25519VerificationKey2018: can be used to sign DID Document updates, Verifiable Credentials, Verifiable Presentations, and arbitrary data with a JcsEd25519Signature2020.
- X25519KeyAgreementKey2019: can be used to perform Diffie-Hellman key exchange operations to derive a shared secret between two parties.

As is the implementation of the IOTA DID method it is understood that it would need integration to be interoperable with other SSI frameworks for reading the DID from the Tangle DLT network. As IOTA Tangle networks are immutable networks, once something is uploaded, it can never be completely removed. This directly conflicts with the GDPR's "right-to-be-forgotten" for any

Personal Identifiable Information (PII). As such, it is not recommended to use the IOTA DID for persons, but only to be used for the Identity of Organisations and Things (and those Things that are not used by an individual).

As this is a big limitation for the majority of ARCADIAN-IoT use cases it is ruled out at this point and so will ARCADIAN-IoT not use this DID Method.

2.1.2.1.2 *Sidetree node for publishing the Public DID Doc using Permissioned Blockchain as the trust anchor*

As ARCADIAN-IoT will also provide a Permissioned Blockchain an option would be to re-use the Permissioned Blockchain to anchor the trust in a distributed Sidetree overlay network.

An open-source implementation of Sidetree that is under development by Transmute Industries is currently being investigated and is available on github [15]. This implements the Sidetree version 1.0 protocol, whose purpose is to create a blockchain based public key infrastructure, where rather than having a central authority that can accept or revoke keys, by having the blockchain act as a immutable witness for registering public keys, anyone can publish a public key that can be used to establish identity. The Sidetree protocol specifies using a CAS and a Ledger to establish a public key infrastructure, where public keys are stored in a Content Addressable Storage, and pointers to that storage are published on a Ledger.

A simple example of this would be a publicly available server, where anyone could upload a public key and an identifier for that public key. However, in essence this sets up a central authority and a single point of failure. So instead, the implementation makes use of a public ledger such as Bitcoin, Ethereum or even a Permissioned Blockchain such as Hyperledger Fabric and uses IPFS as a CAS to point to Decentralized Identifiers and access the Public Keys in the hosted DID Documents. Such an implementation is depicted in Figure 3 shown below.

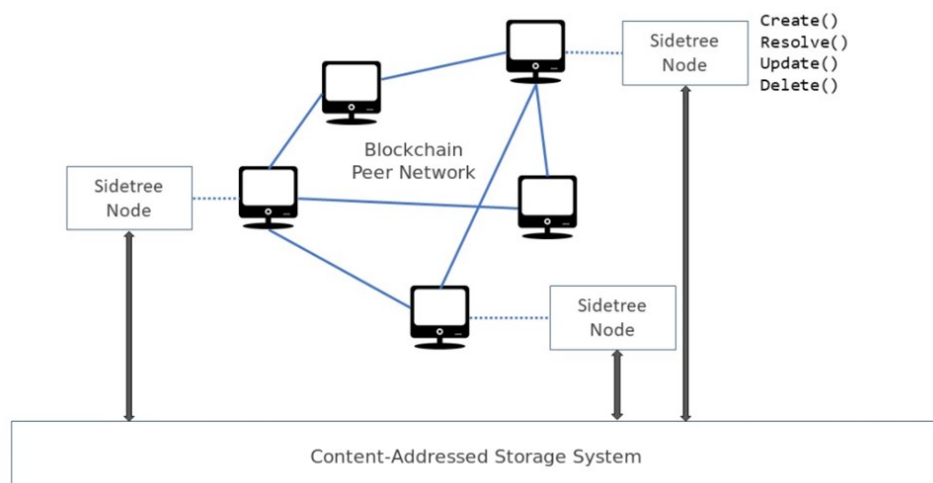


Figure 3 - Sidetree DID Method Overlay network [16]

Architecturally, a Sidetree network is a network consisting of multiple logical servers (Sidetree nodes) executing Sidetree protocol rules, overlaying a blockchain network as illustrated by the above figure. Each Sidetree node provides service endpoints to perform operations (e.g. Create, Resolve, Update, and Delete) against DID Documents. The blockchain consensus mechanism helps serialize Sidetree operations published by different nodes and provide a consistent view of the state of all DID Documents to all Sidetree nodes, without requiring its own consensus layer. The Sidetree protocol batches multiple operations in a single file (batch file) and stores the batch files in a distributed content-addressable storage (DCAS or CAS). A reference to the operation batch is then anchored on the blockchain. The actual data of all batched operations are stored as one. Anyone can run a CAS node without running a Sidetree node to provide redundancy of Sidetree batch files. The Transmute Technologies Sidetree specification [16] is in line with the approach that is being investigated to serve ARCADIAN-IoT and meet the requirement to use the Permissioned Blockchain as a trust anchor for decentralized identity management.

Sidetree supports Create, Update, Recover and Deactivate (CRUD) operations received at a Sidetree API interface. Valid operations are added to the batch writer queue and to the DID cache. Batch writer will then batch multiple Sidetree operations together and store them in Sidetree batch files, over the CAS Interface, as per Sidetree file structure specification.

Next, Sidetree batch file information will be stored into anchor index by a witness function onto the blockchain. The blockchain anchoring system provides a linear chronological sequencing of operations, which the protocol builds on to order DID PKI operations in an immutable history all observing nodes can replay and validate. It is this ability to replay the precise sequence of DID PKI state change events and process those events using a common set of deterministic rules, that allows Sidetree nodes to achieve a consistent view of DIDs and their DID Document states, without requiring any additional consensus mechanism.

The Observer listens for the blockchain events to identify Sidetree operations, then publishes the operations into data structures that can be used for efficient DID resolutions.

2.1.2.1.3 *Permissioned Blockchain DID Methods to store the hash of DID Doc on-chain*

This type of DID Method would rely on a blockchain based VDR to publish a hash of its DID Doc on-chain across the peer nodes of the blockchain network, while the DID Doc itself is stored off-chain in the peer-nodes off-chain datastore. This enables the resolution of the DID to retrieve the DID Docs through any of the peer nodes running the DID:PRIV method smart contract. Only authorized organizations based on the X509 certificates will be able to perform the DID Create/Update/Read/Delete transactions on the blockchain network, and only the controller organization that created a DID would be able to update it. The DID itself is generated with a cryptographic proof with high entropy to ensure the generation of unique decentralized identifiers. The DID method originates from the word “private” as only those with access to the ARCADIAN-IoT permissioned blockchain network and hence its ecosystem will be able to make use of the DID. In the event that a DID is to be deleted, all its DID Doc will be completely removed and only the hash of its DID and DID Doc will remain on-chain. In the strictest sense of the interpretation of the GDPR, then even the hash of the DID could be seen as personal information and because it cannot be removed from the blockchain, then this DID is not recommended for persons, but rather can be used by IoT Devices and Services that are not uniquely linked with person’s.

2.1.2.1.4 *Self-contained DID Methods*

These type of DID Methods setup their own DIDs independently of any 3rd party (be it centralized or decentralized). This type of DID Method is also suitable for most private relationships between people and/or organizations, and it is also cheap and easy to use and well maintained whilst preserving all the security aspects necessary.

- **did:key [17]**

This is a non-registry based DID Method based on expanding a cryptographic public key into a DID Document. This approach provides a simple as possible implementation of a DID Method that is able to achieve many, but not all, of the benefits of utilizing DIDs. While DLT-based DID Methods and more centralized DID Methods provide strong system control guarantees, the general approaches tend to be expensive to setup and operate, whereas use cases requiring DIDs may not require this. For example, a DID that will only be used for a single, ephemeral interaction might not need to be registered, updated, or deactivated. In summary, it is not necessary to store a DID Document associated to this identifier on any DID registry, this is possible due to this method including the public key used in the DID identifier directly. It consists of the did:key prefix, followed by a Multibase base58-btc encoded value that is a concatenation of the Multicodec identifier for the public key type and the raw bytes associated with the public key format. The disadvantage, however, is that it cannot be modified or updated, so if it were to somehow be hacked and another person got control of the corresponding private key it would not be able to be recovered in any way.

- **did:peer [18]**

This is a rich DID method that has no blockchain dependencies and implements a verifiable data registry synchronization protocol between peers. Therefore, it is similar to did:key in that it does not need a distributed or centralized ledger but provides the key information in the did method itself and also a version number so that the DID has an initial inception version 0 when it is created without did doc and then the genesis version 1 adds the did doc.

It seems that parties using peer DID docs could just store raw DID docs. However, any time a DID doc evolves, proof that the evolution is authorized must be found in the DID doc's previous state. If an agent is offline for an extended period (e.g., a phone is lost in the couch cushions for a week), multiple evolutions may have occurred by the time it reconnects and it cannot accept the latest state of the doc without validating the sequence of changes it underwent to get there. Agents must be able to prove to one another that the state they hold is correct. This means that updatable peer DID docs need to be associated with some type of backing storage that adds metadata and history to the simple content of the docs themselves.

Also, as the DID evolves, the subject of a peer DID can update their associated DID document with anyone who knows the DID—one or more agents of the peer(s), or agents of the subject. This operation is more important in the peer DID method than in most other methods, because a loose collection of decentralized peers may include many different views of current state, caused by inconsistent and incomplete connectivity within the peer group.

The DIDCOMM protocol supports the Peer DID protocol and it is also employed in the Hyperledger Aries SSI Framework discussed in section 3.1. However, it is not supported out-of-the-box with other SSI implementations that may use alternative protocols such as SIOP.

2.1.2.1.5 Verification Method Support

The verification method is supported in DID Docs so that a proof can be independently verified. For example, a cryptographic public key can be used as a verification method with respect to a digital signature; so that it verifies that the signer possessed the associated cryptographic private key. This is the basis of all SSI validations for Verifiable Credentials proofs to validate the issuer and the presentation proofs to validate the holder.

The DID Methods support the creation of DIDs that make use of key algorithms used for validating these proofs and it is the SSI framework that must also support them when performing the validation of issued VCs and their presentation.

It is seen that EdDSA ed25519S and ECDSA Secp256k1 are common key signing algorithms supported by the DID Methods verification as analysed in the previous sections, and therefore the SSI Framework under discussion in section 3.1 should ideally support both of these at least for verification and at least one of them for presentation.

Privacy Preserving Verification Methods through BBS+

There is a desirable requirement to support privacy preserving proofs e.g. to support ZKP and other types of privacy preserving measures, as discussed on a leading SSI solution provider's blog [21] and outlined below:

- **Selective Disclosure** – this allows a credential holder to choose which subset of credential attributes are revealed to a verifier, while the rest remain undisclosed.
- **Signature Blinding** – this allows the issuer's signature, which is a unique value and therefore a correlating factor, to be randomized before it is shared with a verifier.
- **Private Holder Binding** – this allows a credential to be bound to a holder without creating a correlating factor for the holder that needs to be revealed upon presentation.
- **ZKP Predicates** – these allow hidden values to be used in operations with a value provided by the verifier. For example, predicates can be used to prove that the holder's bank account balance is above a certain threshold, without revealing the balance.

The BBS+ signature suite has been developed to provide the capability of zero knowledge proof disclosures. However, due to the cryptographic complexity and also so to ease interoperability the BBS+ with LD-Proofs cryptographic specification [6], the ZKP Predicates were dropped to support the other much sought after privacy-enabling features of selective disclosure, non-linkability of VC signatures and credential holders, as described above.

It has been noted that, as BBS+ supports privacy-respecting features indicated above, as well as the use cases where the whole Verifiable Credential has to be presented, it is the common denominator VC format signature suite to support all use cases.

It is therefore a most desirable requirement that the SSI Agents Issuing VCs for persons should support this as well as to support the signature proof validation, and that the DID Methods to be employed in this scenario in ARCADIAN-IoT supports this as a verification method in the DID Doc.

2.1.2.1.6 DID Authentication

It is seen that the DIDCOMM protocol[22] provides intrinsic support for DID authentication which is applicable to devices and systems in ARCADIAN-IoT that are able to support the full Hyperledger ARIES stack.

Where a device is not able to support the full stack (e.g. a constrained IoT Device) or is not needed for a specific scenario it was identified the following mechanisms to provide for DID Authentication:

- Simple DID Challenge / Response Authentication specified by EBSI [48]
- HTTP Signature with signing Key ID provided by the public DID of the client [39]

2.1.2.2 Research findings and achievements

In the previous section it was investigated the state-of-the-art technology in the area of Decentralized Identifiers and touching on their advantages and disadvantages considering the suitability for ARCADIAN-IoT use cases.

Public DIDs for IoT Devices and Service Providers

It was seen that DID methods based on Sidetree with integration into the blockchain is a good candidate for managing public Decentralized Identifiers due to its scalability and trusted distributed architecture based on blockchain and IPFS, and this was subsequently implemented for the first prototype.

However, this was only achieved with private Ethereum and also the use of IPFS was later seen to have issues around not being able to fully guarantee the deletion of any information published on it. It is also a very complex solution with many components and interfaces to be supported, and as such it was decided to support a more light weight DID method that was implemented on the Hyperledger Fabric, as described in section 2.1.2.1.3 in the final prototype P2.

The use of DID:WEB is also seen as a good way for the Service Provider and their services to support DIDs in an independent manner outside the framework as well as to establish trusted registration to the ARCADIAN-IoT Framework.

For supporting SSI Wallets it is preferred the use of peer DIDs to provide an increased level of privacy to the end user.

ZKP support

It was also identified the use of BBS+ signatures to support ZKP privacy preserving measures, such as selective disclosure.

Persons & IoT Devices Authentication to the ARCADIAN-IoT Framework

Authentication of decentralized identifiers is achieved by providing proof of the private key as touched upon in the previous section. Considering the goals of ARCADIAN-IoT for decentralized identifiers to support a standards based Self-Sovereign Identity approach then the authentication of Persons and IoT Devices, identified in the ARCADIAN-IoT framework, is aimed at being

provided by the DIDCOMM protocol provided by the Hyperledger Aries framework (see Verifiable Credentials section 3.1).

Therefore, the authentication of DIDs for persons and devices is not considered under this component, but rather under Verifiable Credentials as it depends on the SSI Agent to prove possession of Verifiable Credentials, based on its private key associated to its DID.

Constrained IoT Devices Authentication to the ARCADIAN-IoT Framework

It is seen that constrained IoT Devices are not able to support SSI stack based on demanding public private key cryptographic operations and an alternative solution is needed based on “DID authentication” being handled by a middleware or IoT GW that acts on the IoT Device’s behalf as described in section 3.1.

Service Provider Authentication to the ARCADIAN-IoT Framework

As regards the authentication of a Service Provider towards the ARCADIAN-IoT framework components, it is proposed to make use of a lightweight public DID authentication outside of the Self-Sovereign Identity standards-based approach. This was not initially foreseen in the KPIs for decentralized identifiers; however, this fulfils another objective for Service Provider services to securely identify and authenticate themselves.

As Service Providers should be able to fully control their own public DIDs in a fully autonomous way, it is seen that the DID:WEB method presented in section 2.1.2.1.1 is the ideal way to do this, with little implementation overhead needed. HTTP Signatures [39] was the protocol proposed to validate all external API calls to the ARCADIAN-IoT framework components as being signed by a registered Service Provider.

Service Provider registration to ARCADIAN-IoT Framework

For ARCADIAN-IoT framework to support SP authentication (see previous subsection) it is needed to first register in the framework as a Service Provider organisation, and this is proposed to be handled in an out-of-bound manner e.g. request by an administrator by company email.

The received information will then be published on the ARCADIAN-IoT’s permissioned blockchain by an organization that has credentials to publish to the blockchain network as described in D3.3 [47].

2.1.2.3 Produced resources

First Prototype (P1) implementations:

- Sidetree node integrated with private Ethereum and hosted by ATOS for integration with the first prototype. This was discontinued in P2.
- DID Peer on the SSI Wallet
- DID:Web for the Service Providers

Final Prototype (P2) implementations:

- Private DID method implemented on ARCADIAN-IoT Permissioned Blockchain.
- DID Authentication based HTTP Signature & Challenge / Response mechanisms
- DIDs supported with BBS+ signatures for selective disclosure.
- Registration of Service Providers DIDs to a Trusted Organization Registry

2.1.3 Design specification

2.1.3.1 Sub-use cases (Recommended)

2.1.3.1.1 IoT-Device's DID Management

The creation and management of public DIDs for IoT Devices, supported by blockchain.

2.1.3.1.2 ARCADIAN-IoT Framework SSI Agent's DID Management

The creation, update, reading and deletion of public DIDs for the ARCADIAN-IoT SSI Agent is supported by the DID:WEB method.

2.1.3.1.3 Service Provider's DID WEB Management

The management and hosting of Public DIDs through the DID WEB method through hosting of the associated DID Docs is fully under the control of the SP organisation that exposes it on a public endpoint.

2.1.3.1.4 Person's DID Wallet Management

The public / private key pair for a user's wallet is created at the initial loading of the wallet app with the public key communicated as part of the Peer DID protocol [41].

2.1.3.1.5 DID Authentication for Service Provider service access to the ARCADIAN-IoT framework

API calls to the ARCADIAN-IoT Framework will be secured by validating them as being signed by organisation decentralized identifiers registered in the framework's Trusted Organization's Registry.

2.1.3.1.6 Register Service Provider to the ARCADIAN-IoT framework

It is needed to register Service Provider organisation's public decentralized identifier in ARCADIAN-IoT. This is handled out-of-band where a SP administrator will email to ARCADIAN-IoT administration their request to add their public Decentralized Identifier to the ARCADIAN-IoT framework's Trusted Organization's Registry.

2.1.3.1.7 Register Service Provider Service to the ARCADIAN-IoT

Once an SP has been registered in ARCADIAN-IoT the SP can automatically register their services in the framework with a public DID associated to the service, and it will be registered with an ARCADIAN-IoT identifier.

2.1.3.1.8 Delete a registered Service Provider to the ARCADIAN-IoT framework

It is needed to delete a registered Service Provider organisation's public decentralized identifier in the ARCADIAN-IoT framework. This is handled out-of-band where a SP administrator will email to ARCADIAN-IoT administration their request to delete their public Decentralized Identity to the ARCADIAN-IoT framework.

2.1.3.1.9 Delete Service Provider Service to the ARCADIAN-IoT

Once an SP Service has been registered in ARCADIAN-IoT the parent SP can request to delete their services in the framework with its associated ARCADIAN-IoT identifier.

2.1.3.1.10 Constrained IoT-Device's DID Management

The creation and management of public DIDs for IoT Devices, supported by DID:WEB method.

2.1.3.2 Logical architecture view

2.1.3.2.1 Public DID published on Sidetree for IOT Devices & ARCADIAN-IoT Agent (P1 Only)

To support public decentralized identifiers in ARCADIAN-IoT framework as discussed in section 2.1.2.1.2 the sidetree specification was implemented for the first prototype P1 as per the following logical design. The use of public DIDs published over sidetree in ARCADIAN-IoT is limited to the ARCADIAN-IoT framework SSI Agent in the first prototype where it acts as an issuer and verifier. In the second prototype IoT Devices will also be issued with public DIDs over sidetree.

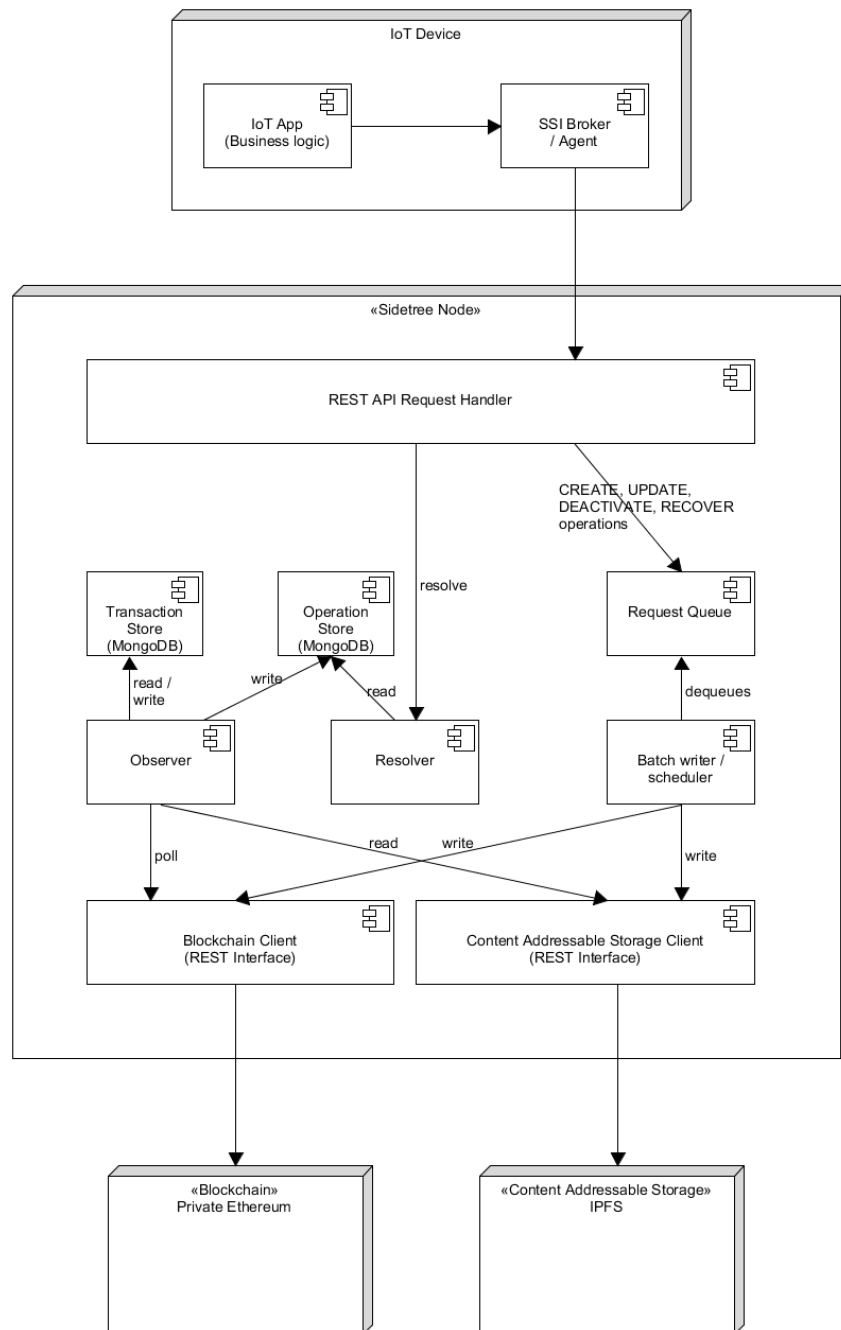


Figure 4 - Public DIDs published on Sidetree Node based on Transmute Sidetree.js [38]

The sidetree node shown components are described below:

Batch Scheduler: This component schedules the writing of new DID operation batches containing CREATE, UPDATE, RECOVER, DEACTIVATE operations.

Observer: This component observes the incoming Sidetree transactions (batch hashes) published on the blockchain and processes them. The observer reads the observed published batch file from the CAS and stores a local copy of it in the Operation store.

Resolver: This component resolves a DID resolution request from the locally stored Operation Store by fetching and compiling all operations that were performed on that DID, as stored in the Operation Store.

Blockchain Client (REST Interface): This component provides a blockchain agnostic interface to provide trust anchor with hash of the batch operation written to the blockchain. The current implementation supports integration with a private Ethereum blockchain network.

CAS Client: This component provides the interface to a hash-based storage interface that sidetree nodes use to publish their DID operation batches so to be retrieved by all sidetree node observer components for network-wide local persistence of all batch operations.

Transaction Store (MongoDB): This component keeps a local record of all transactions.

Operation Store (MongoDB): This component keeps a local record of all batch operations so to aid quick resolving of DIDs and checking of the integrity of the DID operations to reconstruct the latest DID Doc state.

2.1.3.2.2 Public DID anchor published on the Permissioned Blockchain

The public DID for an IoT device is created in the SSI / Broker Agent upon command from the IoT Device business logic, as can be seen from the logical component diagram below. This is then published on the permissioned blockchain on the peer node deployed inside the IoT provider's domain (as provided in D3.3 [47]). This is the Public DID for IoT Devices implemented for the final prototype P2.

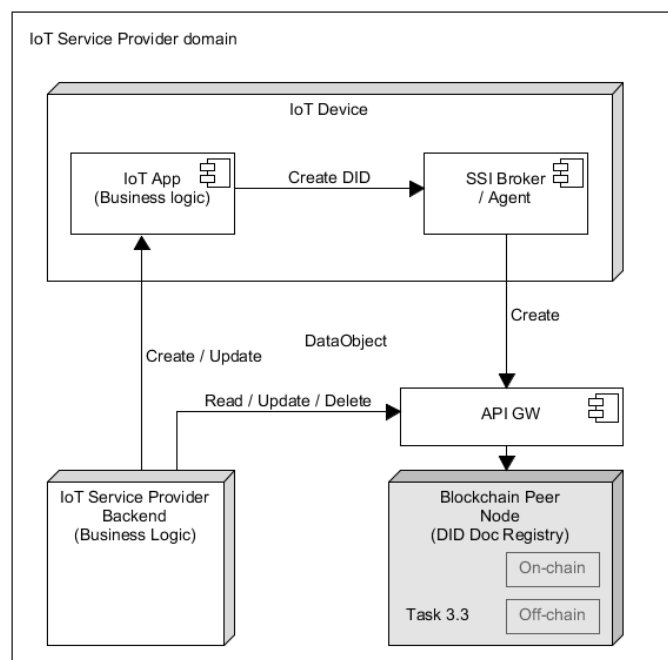


Figure 5 - Public DIDs published on ARCADIAN-IoT Permissioned Blockchain

2.1.3.2.3 Public DID published on DID Web

To support Service Provider (SP) onboarding to the ARCADIAN-IoT framework and authentication to the framework's services, each SP organisation hosts their own DID Doc as per the DID:WEB specification presented in section 2.1.2.1.1. Similarly, constrained IoT Devices will also make use of the DID Web method on the middleware / IoT GW.

The figure below shows DID webs support for allowing Service Providers registration and authentication of service requests in ARCADIAN-IoT framework.

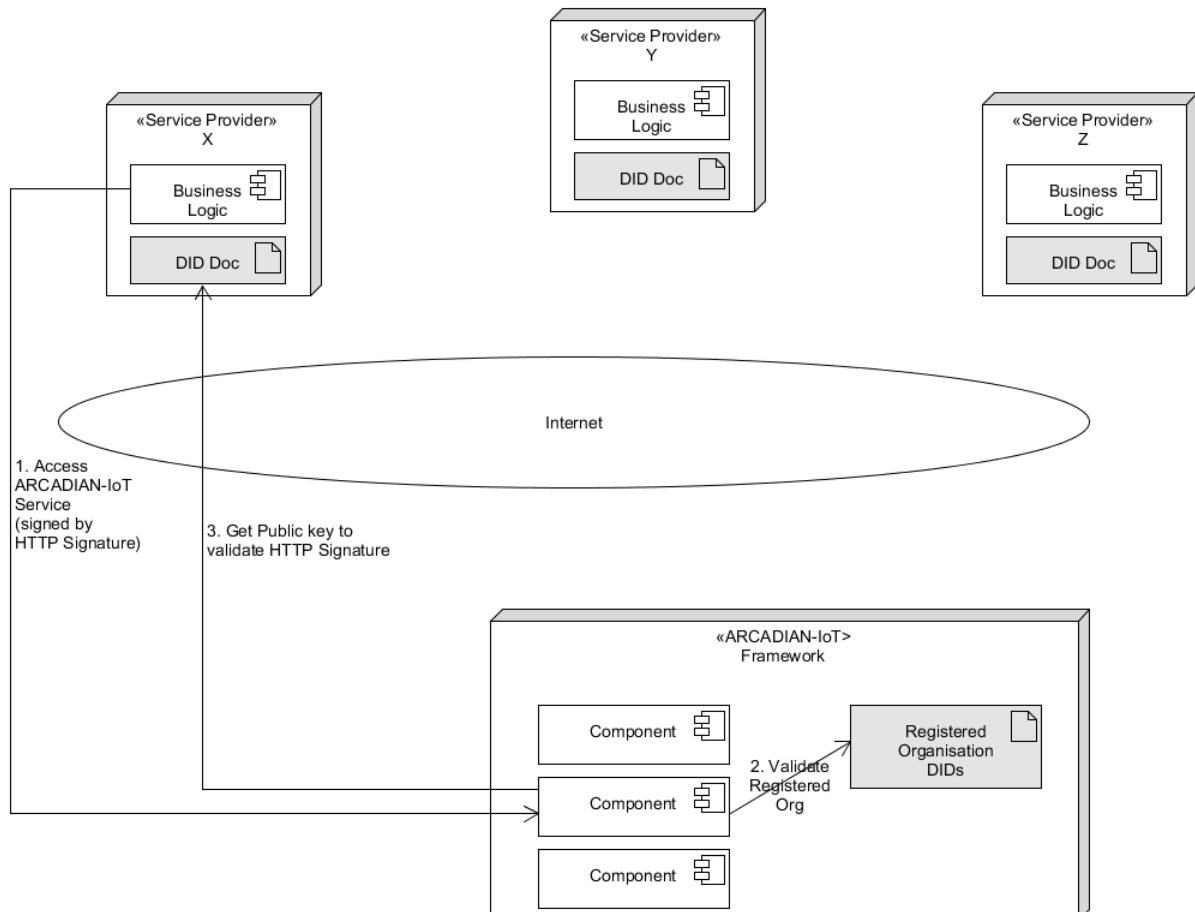


Figure 6 - Public DID:WEB hosted on Service Provider's endpoint to provide service authentication

In the above figure it is seen that each SP exposes its own DID Doc on a public endpoint to publish the public key on a well-known address so that the ARCADIAN-IoT framework can get easy access to it to authenticate any API call it makes to the ARCADIAN-IoT framework from the SP or its services.

A non-normative example is given below for a Service Provider DID, as per the DID:WEB specification [10] and web address it resolves to:

Example SP DID:

did:web:example.com

Resolves to: <https://web:example.com/did.json>

As long as the SP is registered in the Trusted Organization Registry and the API call signature is validated, it will be considered authenticated and authorised to access the ARCADIAN-IoT service. Additionally, each SP service can have its own DID as per the following non-normative example:

Example DID:

did:web:example.com:service:drone_buddy

This DID WEB resolves to the following end point where it can be parsed:

https:// example.com/service/drone_buddy/did.json

An example of the DID Doc exposed on the above address is given below:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1"],
  "id": "did:web:example.com:service:drone_buddy",
  "verificationMethod": [
    {
      "type": "RsaVerificationKey2018",
      "id": "did:web:example.com:service:drone_buddy#ECEA8_A1SlddIICzj4SFS_CJEwxMhZgwgjtO6HPiaqk",
      "controller": "did:web:example.com:service:drone_buddy",
      "publicKeyPem":
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAXZ3Sjwi2hdw80jQO8f/jTPMjtcJUimRR8wHkEPAYQ+0rcCxmV87xK8
kyemr1kns3A6ow8Pf1L0mzBmX4XWpzGPkEtRISBz/I6ICEwdWK/QQSHirDFSGF8Bo6C8EnN+Cwq5ck+Vbr2hzNfY6LQmuy2hv
I5EYLuesMQRd5IBSBLVkyLdlcwrQUUKfT1kxPXS2ILG5GtVU6sWSngZI8JlQLg7pbuzzugCgKVjgmkWwoTiqbFs7jY8P0XZ233t
XeG/KMDMZsPIRdSIBzq5zOBCEnpFzTpbPXUD7VaEXHOS8Ox6EYCuJG2Lt6n+iC8qe5mOH+NU5Wjd1pjipLX1SFkMmTQIDA
QAB"
    }
  ],
  "authentication": [
    "did:web:example.com:service:drone_buddy#ECEA8_A1SlddIICzj4SFS_CJEwxMhZgwgjtO6HPiaqk" ],
  "assertionMethod": [
    "did:web:example.com:service:drone_buddy#ECEA8_A1SlddIICzj4SFS_CJEwxMhZgwgjtO6HPiaqk" ]
}
```

SP DID Authentication

The API call signature makes use of HTTP Signatures specification [39] to authenticate the API. HTTP Signatures enable the ARCADIAN-IoT Framework services to cryptographically authenticate all API calls from SPs while ensuring that the call was not tampered with during transit by making use of digital signature in the HTTP Authorization Header.

Note that HTTP Signatures provide full end-to-end sender authentication and message integrity which is an advantage over mutual TLS authentication that typically has to be terminated at gateway or proxy nodes before reaching its end destination and not all proxies or servers support client certificates.

Finally, as the public key is published in the organisation's DID Doc, the HTTP Signature protocol is essentially authenticating the organisation's DID by proving it has its private key counterpart. For more information on the HTTP Signature client and server implementations see section 2.1.3.4.2.

2.1.3.2.4 Privacy preserving Peer DID for Mobile Wallets

Peer DID conform to the W3C Decentralized Identity specification [40] being able to be used independently of any central source of truth, and are aimed at facilitating private relationships between people, organizations, and things. In ARCADIAN-IoT, Peer DIDs are used to support pairwise DIDs so that a user's wallet can establish connections to entities in a pair-wise fashion as per the figure below.

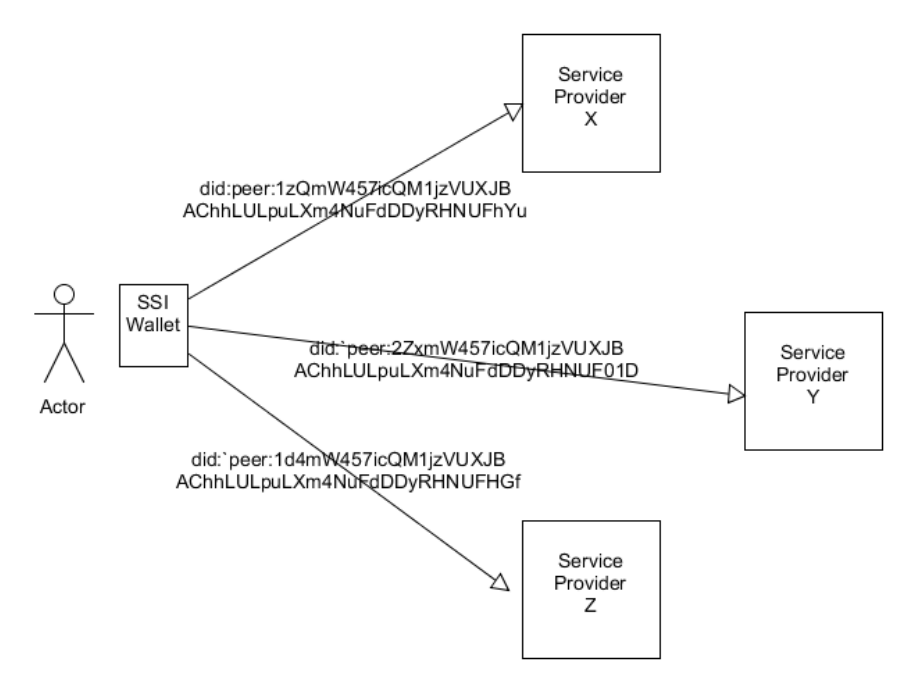


Figure 7 - Privacy preserving peer DID created in SSI wallet per connection

The SSI Wallet implemented in the Ledger uSelf Wallet provided by ATOS, supports pair-wise peer DIDs as per the above figure. More information on Ledger uSelf Wallet is found in section 3.1.

2.1.3.3 Sequence diagrams

2.1.3.3.1 Public DID published on Permissioned Blockchain

Public DID operations are fully implemented and handled by the SSI Agents and their interaction with the blockchain, as shown in the figure below.

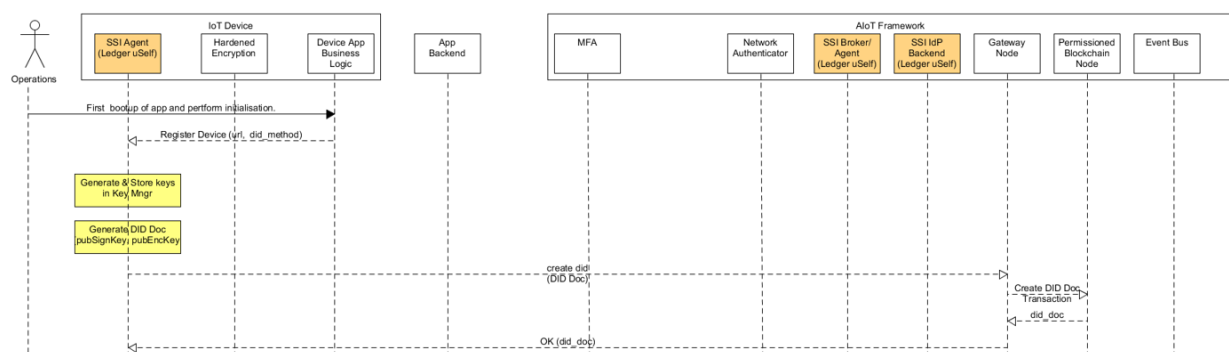


Figure 8 – Public DID published on the Permissioned Blockchain

2.1.3.3.2 Public DID published on DID WEB

There is no sequence flow applicable for this DID creation and management. The organisation administrator can expose the DID Doc end point on the organisation's web.

2.1.3.3.3 Peer DIDs for Mobile Wallet

There is no sequence flow applicable for this DID creation and management. Other sequence diagrams involving the SSI Mobile Wallet and facilitated by the peer DID can be found in section 3.1.

2.1.3.4 Interface description

2.1.3.4.1 Permissioned Blockchain DID:PRIV Operations Interface

The API Interface towards the blockchain supports the DID Operations specified in the sub-use cases section 2.1.3.1.2 (Create, Read, Update, Delete)

For example, the Create DID Operation is supported and is called with the DID and public keys needed to be published with the Decentralized Identity. An example of the result of this operation is given below for a did:priv method created and published on the permissioned blockchain:

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:priv:jtw5KQhnWaoiDqMtJuXvtjnxndng1LAH5c8eUcifmX6eUnfJxg",
  "publicKey": [
    {
      "controller": "did:priv:jtw5KQhnWaoiDqMtJuXvtjnxndng1LAH5c8eUcifmX6eUnfJxg",
      "id": "did:priv:jtw5KQhnWaoiDqMtJuXvtjnxndng1LAH5c8eUcifmX6eUnfJxg#keys-1",
      "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV",
      "type": "Ed25519VerificationKey2018"
    }
  ]
}
```

Figure 9 – Public did:priv method example

2.1.3.4.2 HTTP Signature Interface to support Service Provider DID Authentication

Each SP that wishes to make use of ARCADIAN-IoT framework component services must authenticate as a client conforming to the HTTP Signatures specification [39].

The ARCADIAN-IoT component services that offer an API interface to be called by external Service Providers must authenticate the client call implementing HTTP Signature as a server conforming to the HTTP Signatures specification.

2.1.3.4.3 DID Challenge / Response Interface to support constrained IoT Device Authentication

In reference to the following Figure 10 specified by EBSI [48], the SSI IdP (acting as the RP) performs DID Authentication against the constrained IoT Device DID (acting as Agent) which is actually performed by the middleware IoT GW on behalf of the constrained IoT Device.

Generic DID-Auth Flow

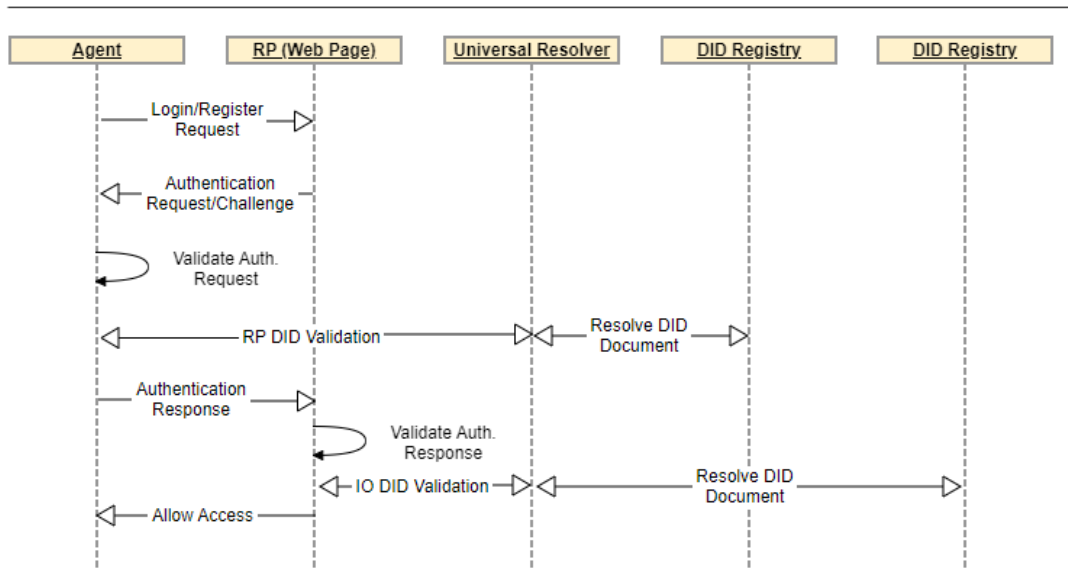


Figure 10 – DID Challenge / Response (Ref: [48])

- The Middleware requests SSI Authentication as per general authentication flow for IoT Devices & persons.
- The SSI IdP Backend looks-up the aiotID & is informed that it is a constrained device.
- The SSI IdP resolves the “subject” DID Web Doc from its DID Web domain and it obtains the DID authentication verification address by adding /did-auth to the resolved url e.g. <https://mediator.box2m.io/123456789A/did-auth>
- The SSI IdP continues to perform DID Auth against the constrained IoT Device DID on the mediator as per the EBSI specification.

2.1.3.4.4 BBS Signature support for Selective Disclosure

The ARCADIAN-IoT project implements selective disclosure (a Privacy Preserving ZKP verification method) provided by the BBS+ signatures supported in the Hyperledger Aries GO framework Ref [49]. The description of its support in ARCADIAN-IoT for persons is described below.

Issuing Credentials

The process of creating verifiable credentials with BBS+ signatures adheres to the standards established by the VC Data Model and BBS+ LD-Proofs specifications. In practice, Hyperledger Aries utilizes the BBS+ Signature Suite 2020 [50], known as BbsBlsSignature2020, to generate verifiable credentials with BBS+ signatures. This suite includes a set of cryptographic algorithms and methods specifically designed for this purpose.

Usage of Credential Schema:

- While the VC Data Model mandates the inclusion of a "credentialSchema" property in zero-knowledge proof systems, BBS+ LD proofs operate differently. BBS+ LD proofs do not require the "credentialSchema" property, thanks to their unique approach to data representation and proof creation.

Example BBS+ Credential:

- A complete example of an issued Verifiable Credential with a BBS+ linked data proof is outlined below. It includes relevant context, type, issuer information, issuance and

expiration dates, and subject data. Importantly, the "proof" section specifies the "type" as "BbsBlsSignature2020," which is essential for its verification:

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/security/bbs/v1"
  ],
  "id": "https://issuer.oidp.uscis.gov/credentials/83627465",
  "type": ["VerifiableCredential", "PersonCredential"],
  "issuer": "did:example:489398593",
  "issuanceDate": "2019-12-03T12:19:52Z",
  "expirationDate": "2029-12-03T12:19:52Z",
  "credentialSubject": {
    "id": "did:example:b34ca6cd37bbf23",
    "personalIdentifier": "X7314321RX",
    "familyName": "SMITH",
    "firstName": "JOHN",
    "dateOfBirth": "1958-07-17"
  },
  "proof": {
    "type": "BbsBlsSignature2020",
    "created": "2020-10-16T23:59:31Z",
    "proofPurpose": "assertionMethod",
    "proofValue": "...",
    "verificationMethod": "did:example:489398593#test"
  }
}
```

Figure 11 Issued Verifiable Credential with BBS+ proof

Presenting Derived Credentials

Deriving Credentials:

- To create derived credentials suitable for selective disclosure, it is essential to follow the BBS+ Signature Proof Suite 2020. This suite defines the cryptographic techniques and processes necessary for generating these derived credentials while maintaining security and privacy.

Disclosing Required Properties:

- When creating verifiable presentations, it is crucial not to inadvertently leak information that would allow a verifier to correlate the holder across multiple presentations. While the VC Data Model suggests that certain properties can be omitted for this purpose, required properties must always be disclosed, regardless of the potential for correlation e.g. "issuanceDate".

Exchanging Derived Credentials:

- When conducting a Presentation Exchange, it is vital to include the "ldp_vp" Claim Format Designation, which signals the use of derived credentials. Within this designation, the "proof_type" property should specify "BbsBlsSignatureProof2020" to align with BBS+ signatures.

Example of Derived Credential for Selective Disclosure:

- An illustrative example of a derived BBS+ credential for selective disclosure is presented below, featuring context, type, subject data, and a "proof" section with the "type" specified as "BbsBlsSignatureProof2020":

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/security/bbs/v1"
  ],
  "id": "https://issuer.oidp.uscis.gov/credentials/83627465",
  "type": ["PersonCredential", "VerifiableCredential"],
  "credentialSubject": {
    "id": "did:example:b34ca6cd37bbf23",
    "familyName": "SMITH",
    "firstName": "JOHN",
    "dateOfBirth": "1958-07-17"
  },
  "expirationDate": "2029-12-03T12:19:52Z",
  "issuanceDate": "2019-12-03T12:19:52Z",
  "issuer": "did:example:489398593",
  "proof": {
    "type": "BbsBlsSignatureProof2020",
    "nonce": "...",
    "proofValue": "...",
    "verificationMethod": "did:example:489398593#test",
    "proofPurpose": "assertionMethod",
    "created": "2020-10-16T23:59:31Z"
  }
}
```

Figure 12 Presented derived Verifiable Credential with BBS+ proof

Verifying Presented Derived Credentials for Selective Disclosure:

It is important to note that when verifying presented derived credentials in Hyperledger Aries base, any blank node identifiers in the credential must be transformed back into their original blank node identifiers before verification.

2.1.3.5 Technical solution

2.1.3.5.1 Deployment architecture view

The final prototype P2 deployment architecture view for supporting IoT Devices with public DID method over the permissioned blockchain can be seen to be overlayed on the logical component view in section 2.1.3.2.2.

2.1.3.5.2 API specification

The SSI Agent supports the creation and resolution of Public DIDs over the permissioned blockchain as per the following example API calls:

```
curl -X POST -H "Content-Type: application/json" -
d '{"didDoc": {"@context": "https://www.w3.org/ns/did/v1", "id": "did:priv:23/11", "public
Key": [{"id": "did:fabirc:123456789abcdefghi#keys-
```

```
1", "type": "Ed25519VerificationKey2018", "controller": "did:fabric:123456789abcdefghi", "publicKeyBase58": "H3C2AVvLMv6gmMnam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
}}}' https://vdrfabric.preprod.ari-bip.eu/store
```

```
curl -X GET "https://vdrfabric.preprod.ari-bip.eu/query?didId=did:fabric:23/11"
```

The API GW interface APIs are specified in section 4.3.3.3 in D3.3 [47].

2.1.4 Evaluation and results

The first prototype (P1) provided:

- A public Decentralized Identifier created by the Sidetree Node rooted on the blockchain.
- A Decentralized Identifier based on DID:WEB.
- A Decentralized Identifier based on DID:PEER communicated in-band between SSI Wallet and remote SSI Agents.

The final prototype (P2) provided:

- A public Decentralized Identifier for IoT Devices implemented on the Permissioned Blockchain supporting Create, Read, Update and Delete operations.
- A code base that supports BBS+ signatures for selective disclosure.
- Registration of Organization DIDs in a Trusted Organization Registry.

The DIDs were validated against the W3C Decentralized Identifier specification [3].

Test validation has been carried out internally by ATOS and component integration and pilot validation is carried out in WP5.

2.1.5 Future work

The DIDCOMM layer 2 protocol supported in ARCADIAN-IoT and discussed in section 2.1.2.1 is an alternative to OpenID SSI protocol stack. Recently GAIA-X has changed its specification from its support of the DIDCOMM protocol to the OpenID SSI stack so to align with the convergence in Data Spaces. Additionally, during the project lifetime EBSI also opted for the OpenID SSI Stack for supporting persons authentication.

Therefore, the primary future work (beyond the ARCADIAN-IoT project) identified will be to evolve the SSI solution to support the OpenID SSI Stack for issuing and presenting W3C Verifiable Credentials.

Additionally, although the ZKP selective disclosure codebase is supported with the BBS+ Keys (in Task 4.1), it has not been able to be verified as this is pending Verifiable Credentials (described in section 3.1) to support this in a new version of its wallet, which is beyond the ARCADIAN-IoT project.

2.2 eSIM – Hardware-based identification and authentication

2.2.1 Overview

2.2.1.1 Description

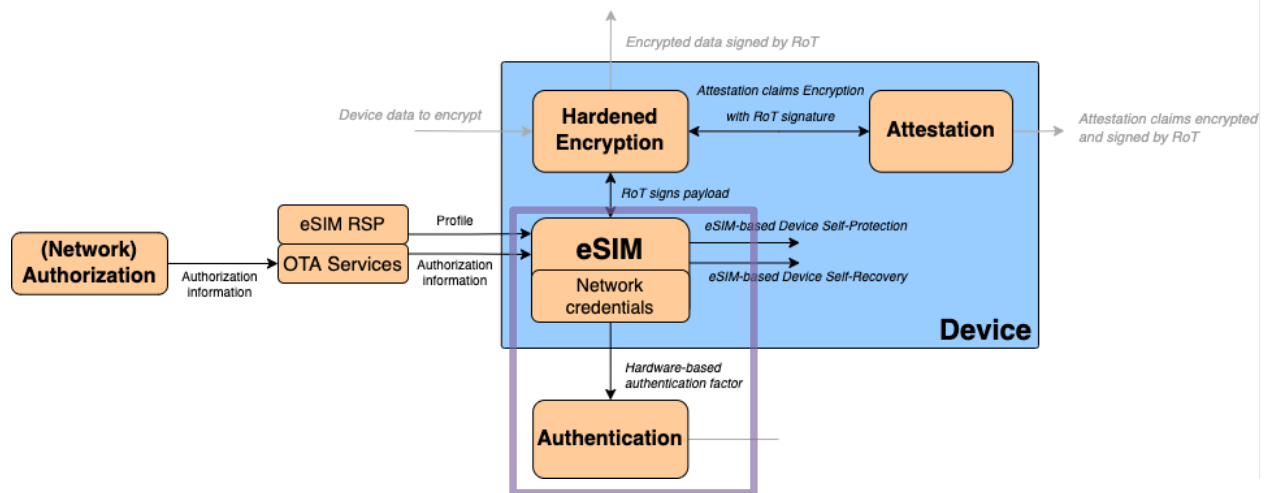
The eSIM is the evolution of the well-accepted and widespread SIM (Subscriber Identity Module) technology to an embedded format with remote provisioning and management capabilities, while maintaining its security processor characteristics. Its novel ecosystem⁷ provides a fully digital management of devices' connectivity and is in itself an enabler of innovation in terms of potential for automation (e.g. for provisioning and management of a large number of devices connectivity profiles according to programmatic rules) and integration with other relevant technologies (e.g. artificial intelligence, or reputation systems, triggering actions in the devices secure element). It also enhances security by design, given that threats related with the use of the secure element in a different device from the one it was provisioned to are almost impossible (the secure element is embedded/soldered at devices hardware).

Particularly, in the context of ARCADIAN-IoT, the eSIM component acts simultaneously as:

- a. **Secure connectivity enabler** for devices and people – enabling the connectivity of the domains' IoT and personal devices through the provisioning of an ARCADIAN-IoT eSIM profile to the eUICC (hardware in the device that receives eSIM profiles).
- b. **Secure Element (SE)** capable of storing identity and authentication credentials at devices hardware level and use them in ARCADIAN-IoT network-based authentication, which is part of the project multi-factor authentication process.
- c. **Root of Trust (RoT)** with ability to contribute to Hardened Encryption and Attestation processes, providing evidence that allows to infer data integrity and trust.
- d. **Local (edge/IoT device) authorization agent** able of receiving trust information about the device where the eSIM is and have specific actions of self-protection and self-recovery.

eSIM is, therefore, a relevant security agent for connected devices with the several roles depicted in **Error! Reference source not found.**. In this section, which belongs to the identity management layer of the project, the eSIM-related work will focus on its role related with the item b. above.

⁷ <https://www.gsma.com/esim/wp-content/uploads/2018/12/esim-whitepaper.pdf>

Figure 13 - eSIM component overall view⁸

ARCADIAN-IoT authentication relies on a multi-factor process to identify and authenticate users and devices (section 2.4). The **network-based authentication**, focused on this section, presents the framework with a novel method to authenticate any SIM-equipped device to a third-party service (running, e.g., in the Cloud) by leveraging cellular networks authentication [56] whose credentials and processes are securely stored at hardware level in the device UICC. The proposed technology is applicable to any SIM form, e.g., SIM, eSIM or iSIM.

The network-based authentication component leverages the standardized and widely used authentication mechanism of network subscribers to cellular network providers, well-accepted as secure for decades. Every device that connects to a cellular network has assigned a unique identity and a set of processes, e.g. of challenge-response between the device and the network; and of authentication and cyphering [57] Furthermore, the communication between the subscriber and the Core Network is protected by the use of a particular licensed spectrum. The well-accepted cellular network authentication processes rely on the SIM, a hardware secure element with computing and storage capabilities, to have stored the mentioned information and processes. In ARCADIAN-IoT we use this authentication process for cellular IoT devices, but also for persons. For using it for persons we assume that their identification will be attached to the one of their personal devices (subscriber information, stored in the secure element of the personal devices, e.g., a smartphone), and that the attachment process (person to personal device) will happen in the registration moment.

There are several motivations to leverage the trust in the cellular network authentication to provide a novel network-authentication mechanism for IoT devices (and persons acting in IoT ecosystems) to third-party Cloud services. It is a zero-touch mechanism that avoids new credential provisioning to the device at manufacturing time, saving considerable effort and costs, and increasing scalability. The credentials used are stored in a hardware secure element, the SIM, well accepted as secure and already present in all cellular devices. Furthermore, this technology avoids the use of other approaches known as prone to attacks, like the use of username/password hardcoded on the devices filesystem, and the use of different hardware secure modules, which would increase the cost, add the provisioning effort previously mentioned, and that is impracticable in many IoT use cases.

Before the project, 1GLOBAL already had a patent [77] of this idea, with an experimental proof of concept assessed as being in TRL 3 at the beginning of the project, as only the critical functionality was experimented in one device and one Cloud service. For example, it was not integrated in any relevant environment, and the concept had no concern of energy efficiency. In ARCADIAN-IoT the goal was to bring it to TRL 6, researching to enhance and demonstrate it in at least two of the IoT domains targeted in the project. Another objective was to make this technology agnostic to

⁸ Diagram from D2.5 depicting eSIM multiple roles in ARCADIAN-IoT

the IoT devices characteristics in terms of processing power, energy consumption and communication protocols used. The technology should be ready for both IoT use cases with devices with high computing capabilities and low battery restrictions, and for IoT use cases with devices with high constraints of energy, computing or communication (e.g. devices whose battery needs to last for 10 years, with computing power just to read a sensor and send its data to a cloud provider once a week).

2.2.1.2 Requirements

The requirements⁹ for ARCADIAN-IoT's network-based authentication component were:

- Leverage proven cellular network authentication procedures and the SIM to **authenticate devices and persons to third party services** (e.g. IoT service providers).
- Have a solution **agnostic to the device characteristics and to the third-party** to which the device needs to authenticate to.

A natural assumption that exists for the use of this component is that IoT and personal devices need to be cellular devices, needing thus to **support a SIM** (any SIM form).

2.2.1.3 Objectives and KPIs

ARCADIAN-IoT's network-based authentication contributes to the accomplishment of the following objectives and KPIs from the project.

⁹ Requirements were updated from the ones described in previous deliverables to better fit the component objectives

KPI scope	
In the context of ARCADIAN-IoT objectives of <i>enabling security and trust in the management of objects and persons' identification</i> , we aim to use the SIM to support a robust identity approach at hardware level; and <i>enable an identification approach for persons</i> , where this component will be joint with 2 other identification approaches for persons in a multi-factor authentication scheme. In this case, the person will be identified through the SIM on its personal device (e.g. smartphone).	
Measurable Indicators	
1. Leverage cellular network authentication processes and SIM credentials in a new zero-touch authentication of IoT and personal devices in third-party services (Yes/No) 2. Number of different devices where the innovation is demonstrated 3. TRL	
Baseline	
1. Experimental proof of concept with 1 IoT device authenticating in 1 public Cloud 2. One test device 3. TRL 3	
Target value	Achieved value
1. Yes, leverage cellular network authentication processes and SIM credentials in a new zero-touch authentication of IoT and personal devices in third-party services 2. At least 2 device types 3. TRL 6	1. Yes. Validated with all cellular devices from 2 IoT solutions from consortium partners, allowing the zero-touch authentication of the devices to their Cloud backend services. 2. IoT devices and personal devices (2 types communicating with 2 protocols: HTTP and MQTT) 3. TRL 5: technology validated in 2 relevant environments (2 IoT solutions from service providers). The objective is to achieve TRL 6 until the end of the project with the technology being demonstrated in relevant environments provided by 2 IoT solution providers from the consortium.

The results above mentioned are further detailed in the evaluation section (**Error! Reference source not found.**).

2.2.2 Technology research

Focusing on the technology research of the network-based authentication component, this section starts by describing the relevant background, moving after to the research findings and produced resources. The content here presented is also part of a recently accepted scientific publication [58].

2.2.2.1 Background

Currently there is a myriad of alternatives for performing IoT authentication, however most of them are not suitable for every IoT device due to lack of energy efficiency, communication overhead, significant storage or computational cost, dependence of dedicated hardware, or security or privacy vulnerabilities [59]. Moreover, current practices still rely on vulnerable approaches like hard-coded credentials (usernames and passwords) [60][61] or in certificate-based mechanisms that tend to be expensive [62], unable to scale due to the provisioning effort needed in manufacturing time [63], or not optimal to use in constrained IoT devices [64].

However, some authentication mechanisms specifically designed for (constrained) IoT have been developed. In [65] an authentication mechanism for Industrial IoT (IIoT) bypasses the constraints of the sensor nodes by using lightweight cryptographic solutions to authenticate the sensor nodes to the gateway nodes that will then aggregate data and send them along to third-party services. In [66] a lightweight authentication scheme for wireless sensor networks is presented relying on

pre-shared keys. In [67] the authors use elliptic curve cryptography to perform mutual authentication between IoT devices and gateway-nodes which provides a secure authentication mechanism to IoT architectures that use gateways. However, all of these solutions impose security and/or computational trade-offs.

The IETF Authentication and Authorization for Constrained Environments (ACE) Framework [68] proposes a relevant lightweight authentication alternative. It is designed to expand the authorization flow of OAuth 2.0 to constrained IoT devices. OAuth 2.0 [69] is an authorization framework that enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf. Two key differences between ACE and OAuth 2.0 are that ACE aims to accommodate the limitations of constrained IoT devices and networks with the use of CBOR (Concise Binary Object Representation) Web Tokens (CWTs) instead of JSON Web Tokens (JWTs), and the use of the Constrained Application Protocol (CoAP) as the application-layer protocol instead of HTTPS. CoAP [70] is an application-layer protocol built specifically to cope with constrained IoT devices and environments, prioritizing the reduction of network traffic. While non-confirmable messages are not used in CoAP, since acknowledgement messages introduced additional overhead to the protocol, they can always be explicitly sent to confirm message delivery. CWTs [71] are a token format that uses CBOR encoding instead of the more traditional JSON encoding. CBOR [72] is designed to minimize communication overhead and encodes information more efficiently than JSON, trading-off human-readability for efficiency. The use of CWTs allows the creation of more compact tokens. By using CBOR and CoAP, the protocol can reduce payload size and communication overhead, improving the efficiency of the use of network resources. In what regards alleviating effort from constrained IoT devices and improve their energy efficiency, some studies propose the offloading of services as a way to enhance energy and performance issues [73][74].

Concerning the security features of cellular networks, these rely on the use of licensed spectrum, subscriber identity modules (SIMs) and security standards [75]. The cellular core network is the network infrastructure positioned between the device and the public internet that comprises, among other functions, the subscriber authentication and service authorization, and sub-network routing services interlinking several nodes that tie together several cellular network types. The authentication mechanisms used in cellular networks are standardized by 3GPP and have been target of continuous improvement over the years with the adoption of the several network generations (e.g. 3G, 4G and 5G). These provide means for a cellular device to prove its identity in a particular network operator, ensuring the confidentiality of the subscriber identity and ciphering the data and signaling [76]. In the client side, each subscriber particular secrets and algorithms that execute the security processes (like the authentication to the network) are stored in a hardware secure element existent in all cellular devices - the SIM.

The background/baseline solution for the network-based authentication component is a previous patent from 1GLOBAL [77]. This work considers OAuth 2.0 to leverage the trust in the cellular core network authentication, to authenticate IoT devices towards third parties (e.g. Cloud services). The motivation for this solution starts with the fact that any connected cellular device has a secure element (the SIM) that is able to securely authenticate the network subscriber to the network provider. For this purpose, each device uses a standardized secure channel to the cellular core network to, using processes and information stored in the secure element, identify itself. The network verifies its identification confirming it as a known subscriber. After, the network authorization policies are applied and the subscriber is able to connect to the Internet. This is done by the functions that have access to the mapping between the requesting client's IP and the SIM credentials (network identifiers). Considering the standard core network authentication and authorization process as background, **Error! Reference source not found.** details the components of the baseline solution towards obtaining and using the ID Token to authenticate and publish data on the Cloud providers. The components are:

- **Network Identity Server / Notarizer:** The network identity server (referred to as *Notarizer*) is responsible for verifying requesting clients' successful authentication in the network core and, if so, issuing ID Tokens that can be used to identify it towards third-party Cloud services.
- **Client:** The client is the constrained IoT device that needs to be authenticated to a third-party using the token issued by the notarizer.
- **Third-Party Cloud service:** The third-party Cloud service is the service to which the client wants to authenticate to. Depicting this solution's flow () it is important to reinstate that before this process starts the client is already authenticated towards the core cellular network.

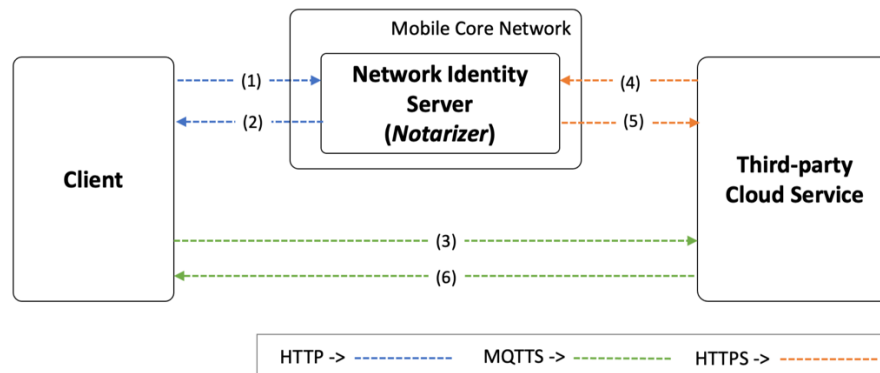


Figure 14 - Network-based authentication component baseline

The flow depicted in **Error! Reference source not found.** starts in (1), when the client asks for an ID Token to the *Notarizer*. Using the client IP, the *Notarizer* verifies in the network core records if the corresponding identifiers are from a known authenticated subscriber. In (2), if the client is a known and authenticated subscriber, the identity server responds with the requested ID Token. In (3), the client uses this token to authenticate itself towards the third-party service. Steps (4) and (5) represent the ID Token verification with the issuer, and (6) is the information of the result of the authentication request. The baseline solution uses HTTP, MQTTS and HTTPS in its flows. An important detail is that while the connection between the client and the identity server uses HTTP, it leverages the standard 3GPP secure connection between the network subscriber and the core network of the provider. Also, considering that the *Notarizer* is deployed in the core network, it is only reachable by that network subscribers. In this first solution, the use of MQTTS between the client and the third-party Cloud service is a requirement from the latter, which limits the IoT device to the use of this protocol with a relevant overhead. The HTTPS communication in (4) and (5) is irrelevant to the IoT device. In what regards the token issued by the *Notarizer*, it follows the specification of the OAuth 2.0 standard. In the baseline implementation it is a JWT containing the mandatory OAuth 2.0 claims and signed by the *Notarizer*. Token validation may be requested to the *Notarizer* or done by the third-party following the OAuth 2.0 standard.

2.2.2.2 Research findings and achievements

The baseline solution already allows to leverage the trust in the cellular network authentication to provide a zero-touch authentication mechanism for IoT devices in third-party Cloud services. It avoids new credential provisioning to the device in manufacturing time, saving considerable effort and costs, and increasing scalability. The credentials used are stored in a hardware secure element, the SIM, well accepted as secure and present in all cellular devices. However, the baseline proof of concept implementation is not optimal in use cases that involve constrained IoT devices. First, the use of HTTP (not HTTP/3), which runs on top of TCP, provides a connection-oriented reliable service at the cost of additional overhead, which may be unbearable by some constrained IoT devices. Also, the use of a JWT is not optimal for devices with low resources.

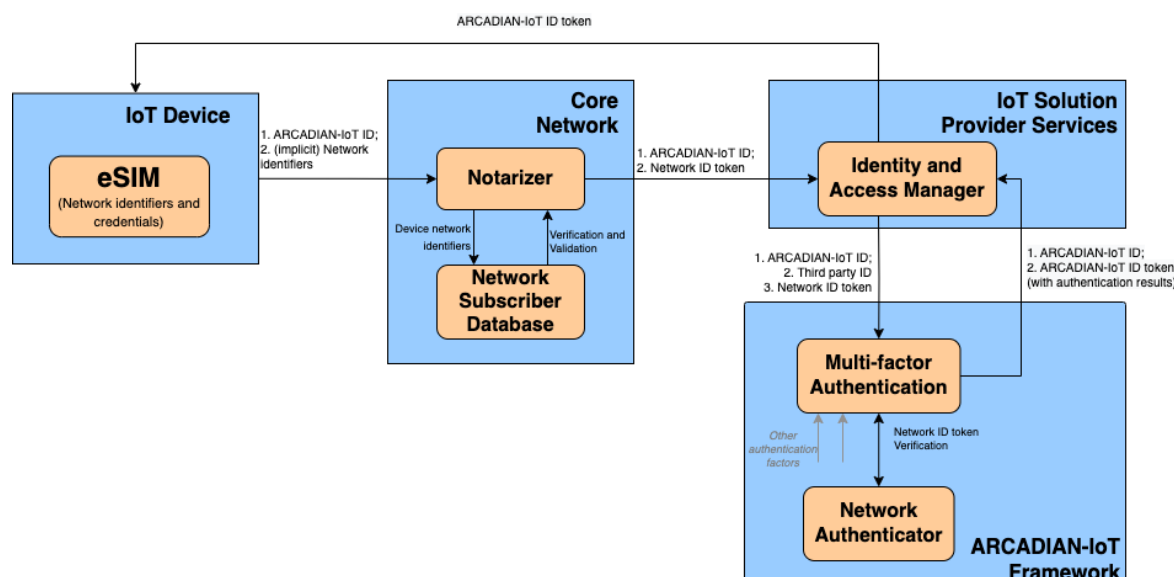


Figure 15 - Architecture of the Network-based authentication in third-party services

In ARCADIAN-IoT, with the research done jointly with IoT solution providers and their IoT use cases, the solution evolved towards being agnostic to the device type, as well as its computing and battery capabilities. The solution also became agnostic to the communication protocol that the device uses and evolved for being integrated in a multi-factor authentication scheme. Particularly, when compared to the baseline, the achievements were:

1. A new architecture that reduces the number of flows executed by the device for authentication – see Figure 14. The solution is now more suitable for use in all cellular IoT devices, including the constrained ones.
2. The *Notarizer* component evolved to incorporate a second role, of protocol translator, beyond the one of ID Server that it already had. Particularly, and according to the specific interests shown by the consortium IoT solution providers, in the scope of ARCADIAN-IoT the devices will communicate either using HTTP or MQTT, and the *Notarizer* converts the communication to HTTPS. Any protocol conversion can be added after without changing the architecture. The main advantage is the reduction of the effort in the device side at no expense of security. This is possible due to the proven security in the communication between the device and the Core Network, and because the *Notarizer* is a Core Network service. Therefore, the communication between the device and the *Notarizer* can be assumed as secure, protected by GSM standards. After, for forwarding of the request through the Internet to the third-party service, the *Notarizer* is able to use any protocol considered secure, according to the third-party capabilities, with no effort for the IoT device.
3. The solution is integrated in ARCADIAN-IoT according to the architecture that can be seen in Figure 14 and for the protocols and ID token requested by the consortium IoT solution providers. It is part of a multi-factor authentication scheme, having a *Network Authenticator*, running on ARCADIAN-IoT Cloud services, for verifying if the network ID token issued by the *Notarizer* is valid and if it matches the ARCADIAN-IoT ID it is supposed to match.
4. It was also concluded that this *Network Authenticator* sub-component would be the right one to, upon the registration of a new entity (person or device), trigger the registration operation not only in its records, but also in the *Network-based Authorization* component. This component lives, as the *Notarizer*, in the Core Network.
5. Other protocols and ID token types were tested and evaluated, specifically following the ACE framework. According to ACE, CoAP and CWT were tested starting with their use in

the baseline approach, adding afterwards the changes for using the *Notarizer* as protocol broker. The gains in terms of effort optimization in the device side were obvious.

Other relevant engineering limitations were found and fixed throughout the project. One example is the *Notarizer* failure rate in issuing ID tokens, that was discovered to be very high in preliminary stages of the project (>20%). In the first prototypes of this component, IoT solution providers noticed and informed that sometimes it was not performing as expected, refusing to provide the requested ID Token, and the authentication failed. This was found to be due to the time to propagate the information about a subscriber authentication in production networks. Sometimes the flow of the device requesting the authentication to the *Notarizer* was faster than the receipt of information that that subscriber had authenticated in the network. In this case, the *Notarizer* would not find the subscriber authenticated in the Subscriber Database, and would refuse to provide the ID Token (as it is supposed to do). As a result, the authentication would fail. An approach that relies on valid network identifiers instead (e.g. the IMSI from a known and valid subscriber) was taken to mitigate the issue.

2.2.2.3 Produced resources

The technical resources produced are the *Notarizer* and the *Network Authenticator* - see Figure 14. The *Notarizer* is deployed in the network provider Core infrastructure and has the role of Network ID Server and communication protocol broker. While thought to be agnostic to the protocol, in this project and according to the IoT solution providers' needs, the *Notarizer* is able to receive HTTP and MQTT requests and translate them into HTTPS. The *Network Authenticator* is deployed in ARCADIAN-IoT Cloud infrastructure. Its main purpose is to verify if the Network ID Token produced by the *Notarizer* is valid and matches the intended ARCADIAN-IoT ID (if it matches the expected person or device ID). These subcomponents are integrated and orchestrated within the ARCADIAN-IoT multifactor authentication.

2.2.3 Design specification

2.2.3.1 Logical architecture view

This section depicts the network-based authentication logical architecture, including its positioning within ARCADIAN-IoT framework.

The key assumption is that the device needs to have a valid ARCADIAN-IoT eSIM (or any other SIM form) provisioned and active in 1GLOBAL's network (the network provider that also provides this component). If it is a valid and active subscriber, it can reach the provider core network services, particularly the *Notarizer*.

After, the beyond state-of-the-art authentication of the device to the IoT Solution Provider Service starts with the device communicating with the *Notarizer* (deployed in the core network), sending it the authentication request with the intended destination. The *Notarizer* uses the device's network identifiers (present in the SIM) to confirm its identity in a subscriber database at core network infrastructure. In case of being a valid and active subscriber, the *Notarizer* crafts a *Network ID Token*, which allows to identify and authenticate the network subscriber, or the device in this case, to the partner/client IoT third-party. This *Network ID Token* (e.g. a JWT) will then be sent to the compliant third-party, who can then confirm its validity by sending it to a *Network authenticator* positioned in ARCADIAN-IoT framework (through ARCADIAN-IoT Multi-factor Authentication). This authenticator verifies if the token is valid (e.g. if it is signed by the *Notarizer* and if it is not expired) and if it (its subject) matches the expected ARCADIAN-IoT ID. The subject is the hash of that subscriber IMSI (International Mobile Subscriber Identity), previously recorded associated with a given ARCADIAN-IoT ID to make this verification possible. If the verification is successful, the *Network authenticator* sends the result to ARCADIAN-IoT *Multi-factor Authentication*, to merge the result with the one of other simultaneous authentication factors

(further details in the authentication section). If all the authentication factors are verified successfully, an ARCADIAN-IoT ID token is generated and returned to the IoT service provider, who returns it to the device for its normal authenticated operation.

2.2.3.2 Interface description

In terms of interfaces, the network-based authentication subcomponents, i.e. the *Notarizer* in the core network, and the network-based authenticator in ARCADIAN-IoT framework, interact mainly with the IoT service provider (with its software at the device) and with the multifactor authenticator, which will be described in the section about authentication. Details about these interfaces can be found in [Error! Reference source not found.](#).

Table 1 - Network-based authentication interfaces

Sender	Receiver	Communication type	Content exchanged
Device (IoT or personal) service	Notarizer (Network-authentication subcomponent)	RESTful API (HTTP)	Authentication request
Device (IoT or personal) service	Notarizer (Network-authentication subcomponent)	Message Publish to MQTT broker (MQTT) ¹⁰	Authentication request
Notarizer	IoT service provider services	RESTful API (HTTPS)	Authentication request w/ network ID token
Multi-factor Authentication	Network authenticator (Network-authentication subcomponent)	RESTful API (HTTPS)	Network ID token + Information about its validity

2.2.4 Evaluation and results

The network-based authentication described in this section is integrated with ARCADIAN-IoT MFA in 2 IoT solution provider services. It is in use for authenticating persons (through their personal devices) and for IoT devices. Therefore, this component contributed effectively to ARCADIAN-IoT objectives, particularly to the one of *enabling security and trust in the management of objects and persons' identification*, where the SIM secure element is used to support a robust identity approach at hardware level. This component also contributed to *enable an identification approach for persons* where, jointly with 2 other identification approaches for persons, it is used in a multi-factor authentication scheme. In this case, the person is identified through the SIM on its personal device (e.g. smartphone).

In what regards this component KPIs, it currently successfully leverages cellular network authentication processes and SIM credentials in a new zero-touch authentication of IoT and personal devices in third-party services. The number of different devices where the innovation is demonstrated includes IoT devices and personal devices (2 types communicating with 2

¹⁰ For simplification, the MQTT broker, which lives in the Core Network as the Notarizer, is considered part of the Notarizer service, to allow MQTT communication between the device and this component

protocols: HTTP and MQTT). At the time of this report the technology is assessed as being in TRL 5, of technology validated in 2 relevant environments (2 IoT solutions from service providers). The objective is to achieve TRL 6 until the end of the project with the technology being demonstrated in relevant environments provided by 2 IoT solution providers from the consortium. This effort is ongoing in the scope of WP5.

During ARCADIAN-IoT progress it was made clear that the proposed solution is very relevant in all IoT authentication scenarios – avoids provisioning effort, leverages secure authentication practices using a secure element already present in cellular devices, it is scalable and has potential to have low costs when compared to the use of other hardware-based approaches. However, for persons, there are use cases that are restrained by this authentication – for now, all the cases where it should be possible for the person to authenticate in a different device than the one it registered with will not be possible with this technology. Moreover, for persons, considering that this technology doesn't demand any user interaction, it should always be combined with other authentication factors, like something that the user knows or a biometrics characteristic, to avoid impersonation if an attacker is using the personal device of the targeted person.

2.2.5 Future work

As future work within the project period, the results will be demonstrated in WP5, within the context of 2 IoT solutions. After the project, the technology will be analysed for commercialization purposes, targeting to improve it, e.g., in the protocols known by the broker, according to the market demands.

2.3 Biometrics

2.3.1 Overview

The biometrics component adds another factor to identify persons entities, relying on their biometrics such as face characteristics. This component will support person verification in two main scenarios. First, face verification will be performed in a photo taken from the frontal camera of a smartphone in order to allow the user to interact with the smartphone app. Second, a more challenging scenario is performing face verification in a video recorded from an Unmanned Aerial Vehicle (UAV). In this scenario, the biometrics component supports facial verification considering operational challenges such as high-altitude recordings, low pixel resolution of the faces recorded, faces from different angles and any disturbance that the UAV may produce over the flight.

In the context of ARCADIAN-IoT, the biometrics component works closely with eSIM and DID to create a multi-factor authentication process to identify the users that requests a particular service.

2.3.1.1 Description

The UWS Biometric component is responsible for receiving a set of photos from the client with the face that constitutes the database to perform face verification, this is named as registration stage. Then, the biometrics component performs identification in two main scenarios:

- Face verification from the frontal camera of the smartphone. The user should be identified every time the app is opened in order to request a service.
- Face verification from a camera attached to a drone. When a user requests an authentication service, the component will process a video feed received from an Unmanned Aerial Service (UAV). Internally, the first stage is to execute a face detection algorithm to search for a face in the video. This is a Convolutional Neural Network-based

algorithm, and it locates and crops the faces that appear in the video. Immediately after, the face verification algorithm will compare the face extracted against the one that the user has previously stored in the database. The database only stores the features of the face extracted from the raw images provided in the registration process, creating then, a safer environment for the user. In addition, together with the identification result, the bounding box that represents the part of the scene where the face has been identified.

2.3.1.2 Requirements

After further analysis, the requirements specified in previous deliverables are modified not because of any deviation but to achieve a higher level of detail. The following requirements are the ones identified for the Biometrics component:

- The system requires several photos of the client's face to perform face verification against a video feed. These images should be shared with the biometrics component in a user registration phase.
- The algorithm requires the reception of an image from the frontal camera of the user's smartphone to authenticate the user.
- The algorithm requires the reception of a video feed coming from the drone in order to allow the biometric algorithm to perform the face identification.
- The algorithm requires HD video resolution.
- The system requires GPU support to execute the algorithm efficiently.
- The algorithm requires adequate lighting to perform accurately.
- The system requires direct communication with the Authentication Component to provide them with the required biometrics results.
- The system requires direct communication with the third-party provider to share the current location of the user in the video.

2.3.1.3 Objectives and KPIs

The following four objectives ensures the quality of the biometrics component:

- 1) Secure storage of the client faces.
- 2) Accurate verification of the users.
- 3) Secure communication with the drone.
- 4) Secure communication with the DGA Service.

With the aim of fulfilling the objectives in the project agreement, the following KPIs will be used to assess and validate the performance of the Biometrics component.

KPI scope	
Low inference time for face verification algorithm.	
Measurable Indicator	
Frames per Second (FPS)	
Baseline Value	
Initial baseline value for the Biometrics asset: 8 FPS	
Target value (M30)	Current value (M30)
16 FPS	18 FPS

KPI scope	
End-to-End speed of the biometrics process.	
Measurable Indicator	
Frames per Second (FPS)	
Baseline Value	
Initial baseline value for the Biometrics asset: 1.5 FPS	
Target value (M30)	Current value (M30)

5 FPS	7 FPS
-------	-------

KPI scope	
High accuracy of the face verification algorithm at close distances (less than 2 meters).	
Measurable Indicator	
F1 score or mAP.	
Baseline Value	
Initial baseline value for the Biometrics asset: 89.05% mAP.	
Target value (M30)	Current value (M30)
Over 90% mAP	90.99% mAP

KPI scope	
High accuracy of the face verification algorithm at far distances (more than 2 meters).	
Measurable Indicator	
F1 score or mAP.	
Baseline Value	
Initial baseline value for the Biometrics asset: 65.23% mAP	
Target value (M30)	Current value (M30)
Over 70% mAP	71.81% mAP

KPI scope	
Reliable verification of the face at close distances (less than 2 meters).	
Measurable Indicator	
False Acceptance Rate (FAR)	
Benchmarking	
0.5% FAR	
Baseline Value	
Initial baseline value for the Biometrics asset: 0.5% FAR	
Target value (M30)	Current value (M30)
Below 0.5% FAR	0.49% FAR

KPI scope	
Reliable verification of the face at far distances (From 2 meters to 15 meters).	
Measurable Indicator	
False Acceptance Rate (FAR)	
Benchmarking (OPTIONAL)	
N/A	
Baseline Value (optional)	
Initial baseline value for the Biometrics asset: 0.5% FAR	
Target value (M30)	Current value (M30)
Below 0.5% FAR	0.46% FAR

KPI scope	
Cost-effective camera and drone platform (hardware only).	
Measurable Indicator	
Euros (€)	
Benchmarking	
N/A	
Baseline Value	
N/A	

Target value (M30)	Current value (M30)
500€	500€

2.3.2 Technology research

Biometric identification has been widely applied to myriad of applications. In ARCADIAN-IoT project, the application of biometrics is focused on drone-based identity management scenarios by exploring AI/ML/data-based approaches in challenging operational conditions (e.g., distance between face and individuals, angle between camera and individual, lighting conditions between camera and individual) while considering necessary privacy preservation.

This drone-based Biometric component will verify a person (focusing on ARCADIAN-IoT users such as the drone pilots for authorisation and the user of the Drone Guard Angel use case service) through analysing their facial characteristics even in challenging conditions introduced by the operation of the drone such as non-frontal face angles and the complex surrounding environments. The technology used in this regard is divided into three different parts: face database, face verification and detection algorithms and machine learning execution platform.

2.3.2.1 Background

1) Faces database

As this component executes the biometrics authentication in two stages: face detection and face verification. Both algorithms should be trained with it corresponding datasets.

- **Face Detection Dataset.** This dataset should cover images of people in different scenarios where the location of their faces is labelled. This first step does not include face verification only detection in the image. The dataset chosen is WiderFace. It is composed of 32,203 images with 393,703 faces labelled within the images. The images are divided into 61 different classes of events, such as: football, festivals, shoppers...
- **Face Verification Dataset:** In order to perform face identification, the biometric algorithm needs to be trained. The process of training is a highly compute-intensive task as it focuses on the extraction of key features of a person's face. As various face features exist, a diverse database containing many people in different environments is vital for the success of facial recognition. Another key factor is the quality of the images in the dataset which is vital. As this system requires the identification of a person from a drone, most of the images should be taken from a similar angle and distance (ideally from a flying platform). The following table presents a summary of SOTA databases:

Dataset	Size (images)	Identities	Drone Friendly	Notes
DroneSURF¹¹	441,000	58 people	Yes	Missing different lightning conditions, distance classification and ethnicities.
LFW¹²	13,233	5749 people	No	Close range images.
YTF	620,952	1595 people	No	Composed of 3,425 videos, most of frames will be exactly alike. Close range images

¹¹ Kalra, I., Singh, M., Nagpal, S., Singh, R., Vatsa, M., & Sujit, P. B. (2019, May). Dronesurf: Benchmark dataset for drone-based face recognition. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)* (pp. 1-7). IEEE.

¹² Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008, October). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In *Workshop on faces in Real-Life Images: detection, alignment, and recognition*.

Color FERET¹³	14,126	1199 people	No	Close range images.
---------------------------------	--------	-------------	----	---------------------

Table 2: Image databases

2) Algorithms

a. Face verification

Once the dataset is collected, it needs to be trained with a face verification algorithm. This is one of the key factors of the whole system as it should take into account several factors to obtain high performance in terms of accuracy and speed. First, the optimal algorithm should be fast enough to achieve real-time performance, and thus, be able to process every frame received from the video feed. Second, the algorithm should be highly accurate to provide credibility to the authentication component. The accuracy should be evaluated in different conditions to distinguish the high and poor performance scenarios. Finally, this accuracy will be obtained in close-range, as the drone will be flying few meters away from the client. In the following table, the algorithms are compared in terms of speed, accuracy. A preliminary study regarding the accuracy at far distances (between 5 and 15 meters) was also executed in order to evaluate the SOTA algorithms when they are tested beyond the capabilities of the dataset (usually images from 1 to 5 meters). In future, distance may also be interpreted as the size of a face in pixels as the lower quality the farther the camera is.

Algorithm	Speed (ms)	Accuracy	Far Distance Performance (2-15m)
ArcFace¹⁴	50	99.84%	64.93%
VGG-Face¹⁵	110	98.95%	74.67%
Dlib¹⁶	10	99.38%	60.54%

Table 3 – Face verification algorithm accuracy

b. Face detection

To conduct face verification, the initial step involves face detection, which is accomplished using a face detection algorithm. It has to be fast to minimize the delays in the face verification pipeline. It also needs to have a high accuracy to ensure precise face capture. Correctly capturing the facial features is paramount for the face verification, as any inaccuracies or omissions may impede the correct verification of the person.

The face detection algorithms have been trained using the WiderFace dataset and subsequently underwent testing on both the WiderFace dataset and the one obtained by UWS. The accuracy on the UAV-UWS dataset is more important, given that it was recorded using an UAV and features numerous images captured from long distances, varying face angles, and under diverse lighting conditions. Moreover, this dataset bears the closest resemblance to ARCADIAN-IoT use case, therefore it is especially significant for assessing the algorithms efficacy.

Algorithm	Speed (ms)	Accuracy WiderFace	Accuracy Far distances
-----------	------------	--------------------	------------------------

¹³ Phillips, P. J., Moon, H., Rizvi, S. A., & Rauss, P. J. (2000). The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions on pattern analysis and machine intelligence*, 22(10), 1090-1104.

¹⁴ Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4690-4699).

¹⁵ Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep face recognition.

¹⁶ King, D. E. (2009). Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10, 1755-1758.

RetinaFace¹⁷	38	81.96%	27.43%
YOLOv7¹⁸	41	65.43%	46.59%

Table 4 - Face detection algorithm accuracy

3) Machine Learning execution platform

Another key factor to be considered is the execution platform of the face recognition algorithm. There are many different frameworks available in the literature although not all of them supports the key aspects of real-time execution of algorithms. In the following table, four main platforms are explored comparing in terms of 4 factors.

Framework	Execution Environment	Code	Deployment	Compatibility (OS)
TensorFlow¹⁹	CPU / GPU	Open Source	Medium	High
PyTorch²⁰	CPU / GPU	Open Source	Medium	Medium
OpenCV²¹	CPU	Open Source	Easy	Medium
SNPE²²	Snapdragon	Proprietary	Medium	Low

Table 5: Machine Learning frameworks

2.3.2.2 Research findings and achievements

The technologies presented in the previous section were compared in terms of datasets, algorithms and execution platforms. These comparisons highlighted the strengths and flaws of each technology/resource available in the state-of-the-art research.

In terms of the face database, DroneSurf is the most promising database to be used for training of the biometrics algorithm. Nevertheless, it has two main flaws: First, the images are just collected from 58 people. This means that the database is not diverse enough and more people should be included. Second, in terms of scenarios, the database was created in good lightning conditions and thereby an algorithm trained with DroneSurf will not perform accurately in a challenging scenario. The main solution to complete this dataset are to manually record and label new videos to include and mix available public datasets. The produced result is exposed in the following section.

From the three face verification algorithms explored in the previous section, just two considers the trade-off between speed and accuracy. ArcFace and Dlib are the most promising algorithms in the literature to begin with. Although they perform accurately at close-range distances, there are flaws when identifying users from far distances needed in our system. Therefore, innovation

Deng, J., Guo, J., Ververas, E., Kotsia, I., & Zafeiriou, S. (2020). Retinaface: Single-shot multi-level face localisation in the wild

¹⁸ Wang, C., Bochkovskiy, A., & Mark Liao, H. (2022), YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors

¹⁹ Martín Abadi *et al.*

TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

²⁰ Paszke, A. *et al* (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems* 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>

²¹ Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

²² Snapdragon Neural Processing Engine SDK, SNPE (2020). <https://developer.qualcomm.com/docs/snpe/overview.html>.

is needed in the face verification algorithm. This produced result is also exposed in the following section.

The two face detection algorithms explored demonstrate comparable inference times. However, it is worth noting that this speed falls short of the pipeline's requirements. In terms of accuracy, RetinaFace excels on the WiderFace dataset, but struggles with the UAV-UWS dataset. YOLOv7 perform poorly in the WiderFace dataset but shows some improvement at far distances, though not to a remarkable extent. This highlights the need for innovation in face detection algorithms. In the following section, the algorithm developed will be introduced and detailed.

In order to execute the face verification algorithm, TensorFlow and PyTorch are the most promising ML platforms. These two platforms have been determined to be the most suitable ones as they are both open source and GPU compatible, besides, these two frameworks accept the development of other models such as: face detection, face alignment and even the implementation of super resolution techniques. Both platforms provide libraries to use their framework for different languages (C, C++, Java, Python), nevertheless, the biometrics component is only implemented in Python.

2.3.2.3 Produced resources

- Dataset Resource

The listed SOTA datasets have some drawbacks such as the images are not classified by distance, or they do not cover low lighting conditions, therefore, UWS is collecting a dataset from far distances between 2 meters and 20 meters. This dataset tries to cover different flying altitudes and distances from the faces recorded. Besides, this dataset will convert a wide range of different scenarios in low-lightning conditions. This is one of the most time-consuming tasks in the process because it needs to ask for permission to fly drones, the time taken to record each volunteer face and manual labelling of these faces. In addition, this is a continuously evolving process as new data is being added periodically.

Dataset	Size (images)	Identities	Drone Friendly	Notes
UAV-UWS Dataset	7500	21	Yes	

Table 7: Dataset specification

- Algorithms

- Face verification

As previously presented, the listed face verification algorithms are not prepared for far distance performance. These are usually enhanced for close range verification which is considered the distance from the user to the camera when a “selfie” is being taken. One of the main outcomes of this work is the creation of an algorithm named as “UWS Model” that achieves the trade-off performance in terms of speed and accuracy required for face verification from far distances. This model takes as a baseline ArcFace algorithm which provides an acceptable but non-sufficient trade-off for the Arcadian-IoT framework.

This outcome is still under development, but it has achieved 55.6 ms and 71.81% mAP of accuracy at far distances (up to 15 meters). UWS Model is 6% more accurate at far distances than standard ArcFace while improving the speed.

Face verification Algorithm	Speed (ms)	Accuracy	Far Distance Performance
<u>UWS Model</u>	<u>55.6 ms</u>	<u>71.81%</u>	<u>Improved performance.</u>

Table 8: Algorithm accuracy

- Face detection

As previously discussed, both of the presented face detection algorithms exhibit limitations in terms of speed and performance at far distances. In response to these challenges, a new face detection algorithm has been developed named as “UWS-YOLO” that achieves high speed while improving the accuracy at far distances. This model takes as a baseline YOLOv7 algorithm which provides acceptable results for testing, but not enough for the Arcadian-IoT framework requirements, therefore the need of improvement.

Our algorithm has achieved a remarkable inference time of 20 ms, which is twice as fast as the other face detection algorithms. Furthermore, it has achieved an accuracy at far distances (up to 30 meters) of 59.29%. Our face detection algorithm is 13% more accurate than standard YOLOv7 in the UAV-UWS dataset and twice as fast.

Face detection Algorithm	Speed (ms)	Accuracy Far distances	Far Distance Performance
<u>UWS-YOLO Model</u>	<u>20 ms</u>	<u>59.29%</u>	<u>Improved performance.</u>

Table 8: Algorithm accuracy

- Machine Learning execution platform

Over the biometrics pipeline many tasks are executed in sequential and parallel order. In order to execute this pipeline, the final execution platform deployed is based on the combination of two different frameworks. From the best of our knowledge, the deployment of OpenCV for video and image processing in combination with TensorFlow as a machine learning framework provides the best results in terms of execution speed. The following section defines how both frameworks are deployed together to achieve better results instead of just deploying one of them for every task.

Framework	Execution Environment	Code	Deployment	Compatibility (OS)
TensorFlow	CPU / GPU	Open Source	Medium	High
OpenCV	CPU	Open Source	Easy	Medium

Table 9: Deployed Frameworks

2.3.3 Design specification

This subsection presents the technical overview of the biometrics component.

2.3.3.1 Logical architecture view

Figure 20 presents the logical architecture of the biometrics components including some of the remaining components from Arcadian framework that has an influence in the behaviour of the biometrics component. Note that for brevity ARCADIAN-IoT is referred to as AIoT in the figure. Four main interfaces are presented: three with third party provider and one with the multi-factor authentication component. Further detail of these interfaces is explained in subsections: 2.3.3.2, 2.3.3.3 and 2.3.3.4.

Besides, the Biometrics component performs multiple computationally expensive tasks in parallel. Five stages are executed by the component to produce the results regarding the authentication of the user. Figure 16 presents the five stages that the images or frames are processed when received from the third-party service to provide authentication results.

- 1) The images received are pre-processed in order to prepare them for an object detection algorithm. In this stage, two phases are executed in OpenCV framework: scale and colour scale. The image is rescaled from 1280x720 pixels to 608x608 pixels. Then, the colour space is changed from RGB to BGR.

- 2) The face detection algorithm will provide the coordinates on the image where the faces are located. The algorithm deployed is named as UWS-YOLO and it is executed with TensorFlow framework. The results in terms of pixel coordinates are fed into the next step.
- 3) This pre-processing stage prepares the images from the original image to the face verification model. First, the face is cropped from the original image. Second, the face is rescaled to 112x112.
- 4) The face verification algorithm provides as a result what is usually named as “embeddings”. This embedding could be defined as the mathematical features extracted from the face provided as input. The execution of the face verification model is performed over TensorFlow framework.
- 5) The final stage obtains the distance between the embedding obtained in the face verification process of the video from the UAV and the embedding obtained from the original face of the person that was registered. A similarity index calculation is performed using cosine distance to determine if the distance of both embeddings is close enough to verify the identity of the person that requested by the authentication service.

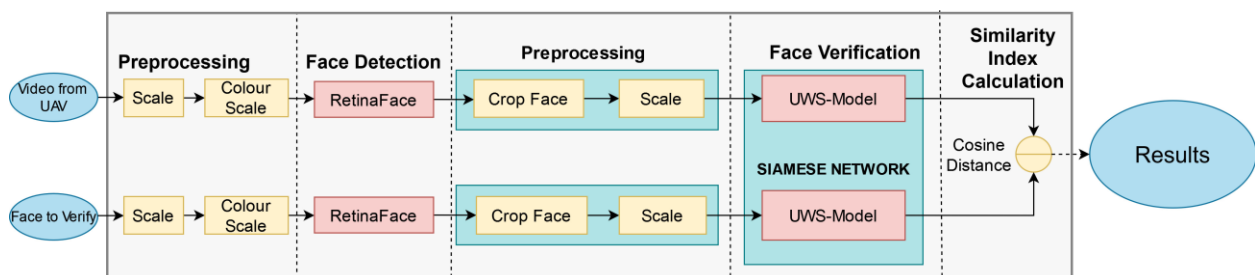


Figure 16 - Logical process view of Biometrics component.

2.3.3.2 Sub-use cases (Recommended)

The Biometrics component conceives four different sub-use cases in order to have a successful interoperability with other components or third-party service.

2.3.3.2.1 Person Registration

Person registration is the first use case required in order to store the face characteristics of the person that will request a biometrical authentication. Two values are stored in the biometrics component database: Arcadian-IoT identifier and the face features of the person. Other components that intervene in this sub-use case are: Authentication, Identity Provider and a Third-Party service.

2.3.3.2.2 Person Update

Person update is an optional sub-use case where the person can update their face characteristics by taken and sending new images from their face. Third-Party service is the only entity that intervene in this sub-use case.

2.3.3.2.3 Person Delete

Person delete is an optional sub-use case where the person can remove their information from the biometrics component database. The information deleted are the Arcadian-IoT service and the features of the face stored. The only entity that intervenes in this case is the Third-Party service.

2.3.3.2.4 Person Authentication

Person authentication use case may be divided in two scenarios:

Authentication request from a personal device where a photo is taken in that moment from the frontal camera of the personal device. This image is then verified against the face characteristics previously stored in the registration sub-use case. The components taking place in this scenario are Authentication component and Third-Party service.

Authentication request from a drone video. In this scenario, a video feed is streamed from a drone and the person being recorded is the one to be verified. The frames from the video received are decoded and processed in order to apply verification against the face characteristics previously stored in the registration sub-use case.

2.3.3.3 Sequence diagrams (recommended)

The sequence diagrams of the Biometrics component are described in the following figures that are correlated with the main use cases: registration, authentication from smartphone (image) and authentication from a drone (video). The communication channels are based on AMQP as explained in API specification subsection.

Registration is presented in Figure 17, the Third Party Service requests to the IdP the creation of an ID for a new user. The ID is received by the Third Party Service and sends it along with the faces of the user to the biometrics component. The features of the faces will be extracted and stored in the biometrics database using the ID previously created. If the process has finished successfully, an ok message will be sent back to the Third Party Service.

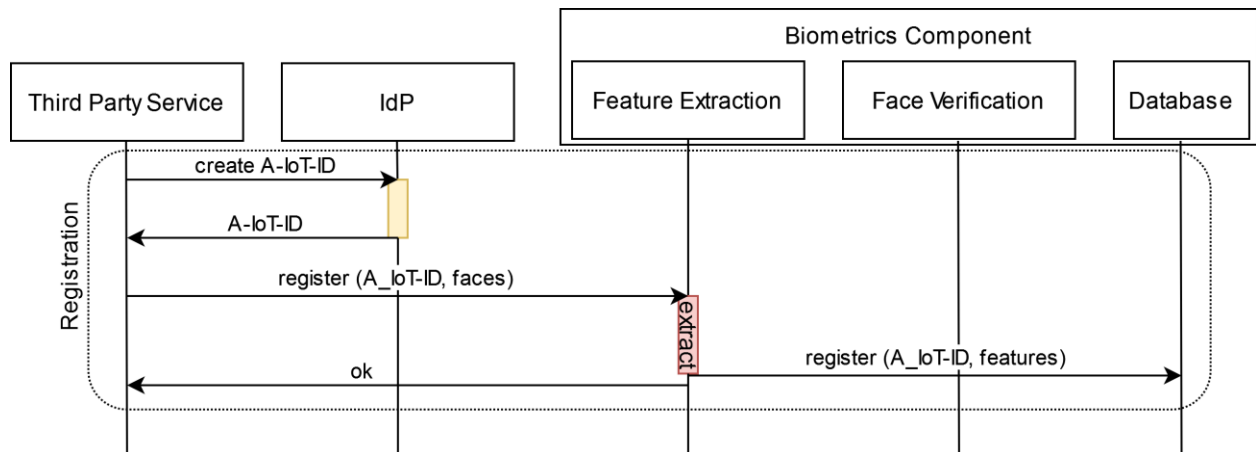


Figure 17 - Sequence diagram for registration use case.

Authentication from smartphone is presented in Figure 18. The Third-Party Service will send a message requesting a face verification to the MFA. The message contains one image to perform a single verification to gain access to open the smartphone app and the A-IOT ID. The MFA manages the authentication process by requesting the identity information of the user to the biometrics component. The Feature Extraction receives the message and requests the features to the database of a user using the ID provided. This user has to be registered previously in the registration process. If the component has received only one image, the features of all the faces in the image will be extracted and the face verification will verify if any person of the image is the one previously stored in the database. The result will be sent back to the MFA.

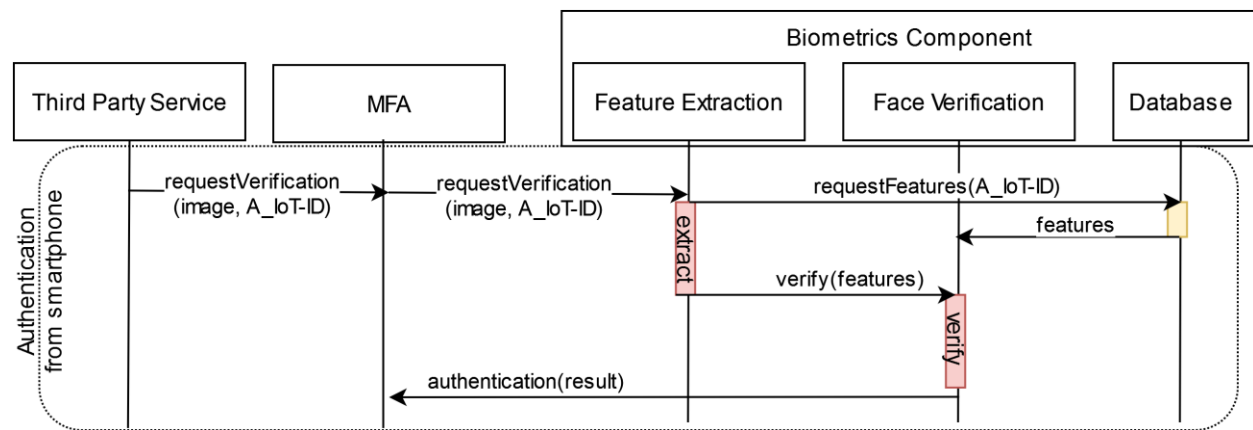


Figure 18 - Sequence diagram for person authentication from a smartphone.

Authentication from a video recorded by a drone is presented in Figure 19. The Third Party Service will send a message requesting a video verification to the Biometrics Component. The message contains a URL to a video streaming to perform face verifications and the A-IoT ID of the person. The MFA forwards this message to the Biometrics component. Then, the video can be requested from the URL provided. The Feature Extraction receives the message and request the features to the database of a user using the ID provided. This user has to be registered previously in the registration process. The video received is decoded and each frame will be verify in order to perform a biometrics authentication of the user. The result will be sent to the MFA.

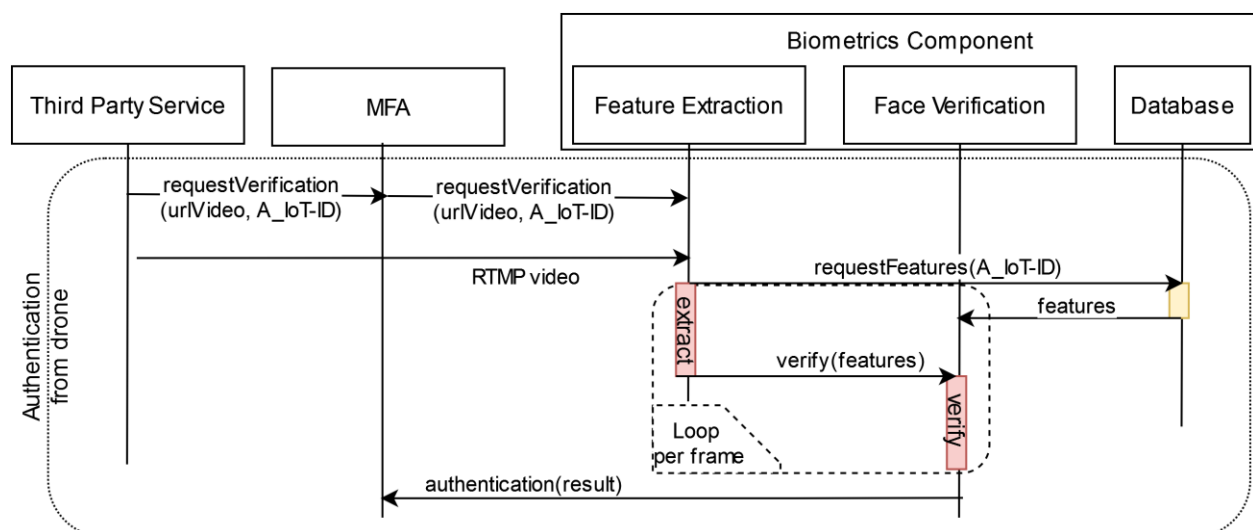


Figure 19 - Sequence diagram for person authentication from a video being recorded by drone.

2.3.3.4 Interface description

At this stage of the project there are three different interfaces that communicate different components and third-party services. These interfaces are directly related to the sub-use cases previously defined.

2.3.3.4.1 Person registration, update, delete

For person registration, update and delete sub-use cases, the Biometrics component is communicated with the third-party service. This communication is performed over AMQP through RabbitMQ software. When a final user wants to register/update/delete their information into the biometric component (A_IoT ID and images), the third party service publishes a message into the RabbitMQ exchange. The biometrics component which is already subscribed will receive the

information and after executing the task requested, it publishes another message which the response status. The AMQP API is further defined in section 2.3.3.5.2.

2.3.3.4.2 *Person authentication*

As explained, the authentication request is divided into two scenarios. First, for the authentication from a personal device, this is performed over a REST API. Second, the interface created for the authentication from a video streamed from a drone is performed also over RabbitMQ.

The description of each interface is defined in subsection “API specification”. The AMQP and REST API is further defined in section 2.3.3.5.2.

2.3.3.5 Technical solution

2.3.3.5.1 *Deployment Architecture View*

Figure 20 presents the architecture view of all the components that has influence in the previous defined use cases: person registration, update and delete and person authentication.

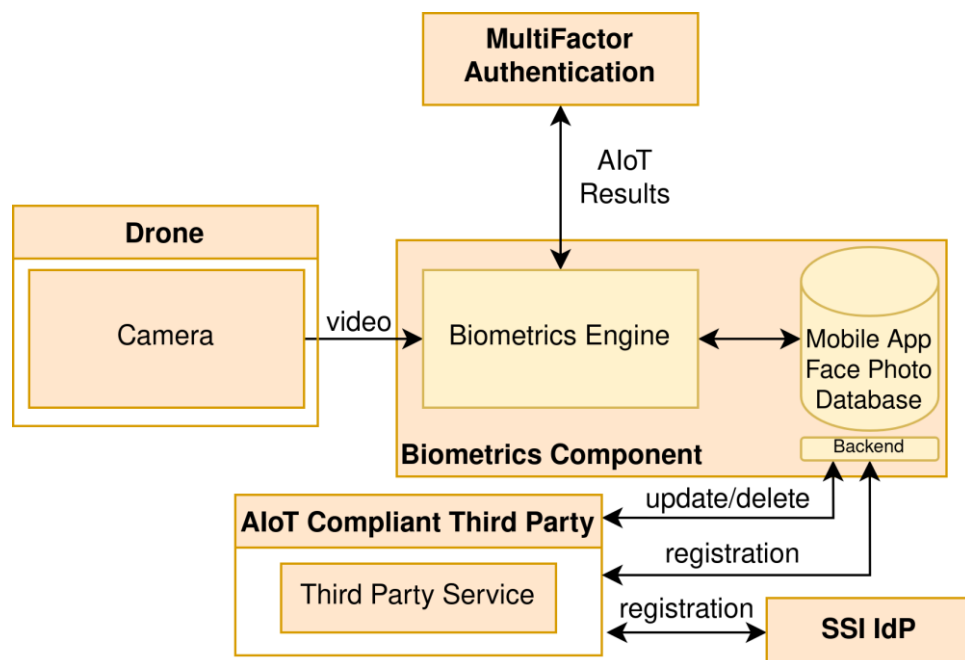


Figure 20 - High-level architectural view of the biometrics component and other components that have relation with.

2.3.3.5.2 *API specification*

The API specification for AMQP is described in the following picture:

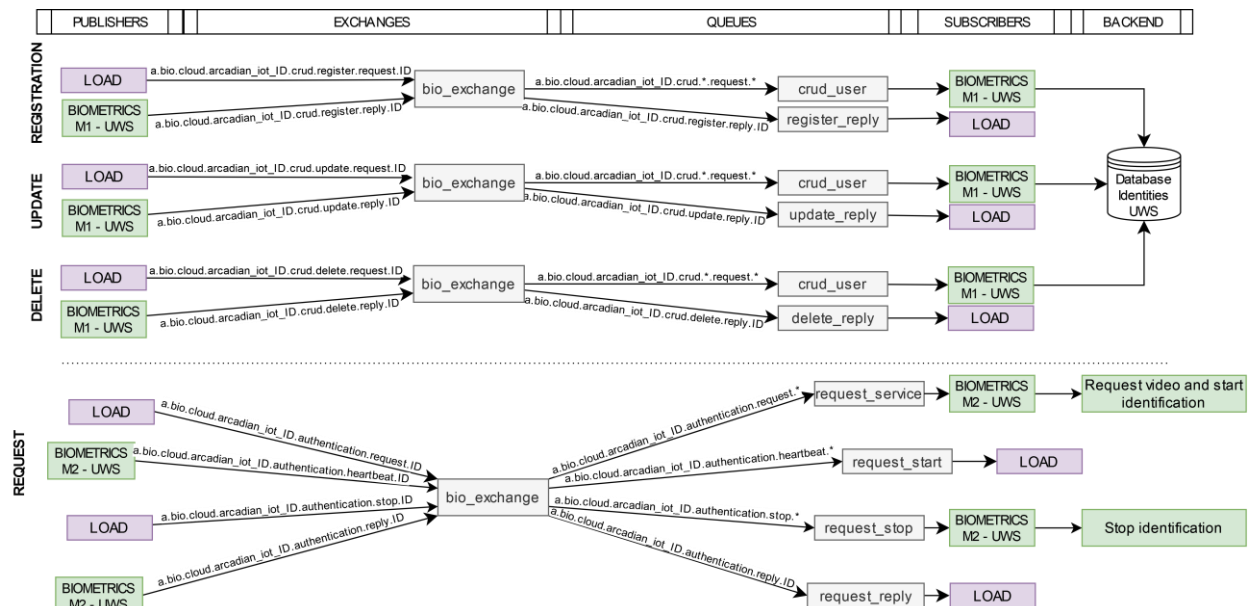


Figure 21 – AMQP API specification for biometrics messaging

The API specification for REST is described in the following table, the biometrics component receives a single image to from the multi-factor authentication in order to verify the people's faces. This authentication is only from the smartphone's camera and will only capture one image.

Request	Reply
POST IP:PORT/authenticate/ Headers: X-AIOT-AUTH-DID: DID Body: {'BiometricsImage:' Image <Base64>}	Error 400 if error in request. Success 200 with body: { "result": "Code with result of the authentication" 0: Authentication Complete 1: No faces detected 2: More than one face detected 3: Other error" "verified": "Boolean (True or False)" }

Figure 22 – REST API specification for Biometrics component

2.3.4 Evaluation and results

The evaluation of our face verification model (UWS model) has been made using the UWS dataset, that contains faces recorded from an UAV at different distances. At far distances (up to 15 meters) our UWS model has achieved 71.81% mAP (KPI: 70% mAP) of accuracy at 18 fps (KPI: 16 FPS), 55.6 ms. At close distances the accuracy achieved is 90.99% mAP (KPI: 90% mAP). All these results have been achieved with a cost-effective platform (drone and camera) of 500€ (KPI: 500€).

2.3.5 Future work

As all the established KPIs have been successfully achieved, the next steps will be set towards the integration and successful validation of the Biometrics component within the Arcadian-IoT framework and Domain A use cases. To ensure the continuous evolution and effectiveness of the Biometrics Authentication system, several areas of technical future work can be explored. First,

improvement the scalability of the system by implementing a distributed architecture that leverages multiple GPUs to handle concurrent video streams, authentication, and registration requests. This can significantly increase the system's throughput and responsiveness, making it capable of handling a larger number of users and devices simultaneously.

Another topic to focus on is the enhancement of the machine learning model. Biometric recognition models benefit from regular updates to adapt to changing user profiles and new attack techniques. Establish a process for periodically retraining and updating the machine learning models used in the system to ensure its long-term accuracy and reliability.

2.4 Authentication

2.4.1 Overview

2.4.1.1 Description

ARCADIAN-IoT authentication relies on a multi-factor authentication (MFA) process to identify and authenticate persons and devices in IoT service providers' services. The targeted input factors are other components from the framework, specifically:

- a. Decentralized identifiers (section 2.1) / verifiable credentials (section 3.1).
- b. eSIM hardware-based/network-based identification (section 2.2).
- c. Biometrics identification (section 2.3).

The main objective of ARCADIAN-IoT authentication component is to be the orchestrator of multiple authentication factors, supporting a robust authentication mechanism for the mentioned entities (persons and IoT objects) in ARCADIAN-IoT third party services. The MFA outcomes will also feed the Behaviour Monitoring component and any other that needs to acknowledge entities authentication results, fostering an integrated cybersecurity platform able to propagate relevant threat information and act with reduced human intervention.

2.4.1.2 Requirements

The main requirements for the authentication component are the following²³:

- **Authenticate persons:** In ARCADIAN-IoT, persons should be able to be identified and authenticated in IoT service providers' services using (1) a decentralized identification approach, (2) a hardware-based identification, and (3) their biometrics characteristics.
- **Authenticate devices:** Devices should be able to be identified and authenticate in compliant ARCADIAN-IoT services using (1) a decentralized identification approach and (2) a hardware-based approach.

2.4.1.3 Objectives and KPIs

ARCADIAN-IoT's MFA component contributes to the accomplishment of the following objectives and KPIs.

²³ Requirements enhanced since the last public deliverables according to the research

KPI scope	
In the context of the objectives of <i>enabling security and trust in the management of objects and persons' identification</i> , the component aims to support at least 2 robust identity mechanisms for devices ²⁴ ; and at least 3 multiple simultaneous identification approaches for persons.	
In this sense, the main scope of this component relates with developing a novel MFA technology joining hardware-based identification, with decentralized identification and biometrics.	
Measurable Indicator	
1. Number of simultaneous different identification factors for persons 2. Number of different identification factors for devices 3. Number of devices used simultaneously in a person's authentication	
Target value	Achieved value
1. At least 3	1. 3
2. At least 2	2. 2
3. 2 (requirement from a demonstrator IoT solution)	3. In WP4 was developed the technology for using 2. Integration and demonstration to happen in WP5.

2.4.2 Technology research

2.4.2.1 Background

NIST defines MFA as “a security enhancement that allows to present 2 pieces of evidence – your credentials – when logging in to an account”²⁵. MFA schemes are getting common to face the issues of traditionally used authentication processes like the use of username and password. As passwords are hard to remember, people tend to use the same in many different digital services, which is well-accepted as a poor practice. To overcome this security issue, MFA schemes are being adopted, especially in online services that deal with sensitive information like e-banking, or professional shared digital services. In MFA schemes, the most common kinds of factors are²⁶:

- Something the person knows, like a password or a PIN.
- Something that the person has, like a smartphone or a secure USB key.
- Something that the person is, like a fingerprint or a facial recognition.

Time and location can be added to these factors to strengthen the credentials verification process²⁷. These factors relate directly with the ones existent in ARCADIAN-IoT. Even the time and location are considered relevant for the scenario of a person authentication using 2 devices for identification as will be explained after.

Usually, for persons, an MFA scheme relies on requesting the person to provide information that allows to validate the several factors, step by step (being this process also known as step-up authentication²⁸). To avoid the burden that may relate with a lower system usability, many systems^{Error! Bookmark not defined.} just requests a second factor for specific functionalities, e.g. to change a password.

In what concerns MFA for IoT devices, this is still quite uncommon. Its application for admin access to IoT devices is considered very relevant²⁹, however this is, again, person authentication.

²⁴ Services identification will be performed only with decentralized identifiers, not in a multi-factor authentication scheme, because the other ARCADIAN-IoT identification factors (eSIM/hardware based and biometrics) don't apply to services

²⁵ <https://www.nist.gov/blogs/cybersecurity-insights/back-basics-whats-multi-factor-authentication-and-why-should-i-care>

²⁶ <https://support.microsoft.com/en-us/topic/what-is-multifactor-authentication-e5e39437-121c-be60-d123-eda06bddf661>

²⁷ <https://www.techtarget.com/searchsecurity/definition/multifactor-authentication-MFA>

²⁸ <https://auth0.com/blog/what-is-step-up-authentication-when-to-use-it/>

²⁹ <https://blog.nordicsemi.com/getconnected/multi-factor-authentication-for-iot>

The same source references the relevance of MFA for ensuring the trust of the devices in a network, referring to the ease of adding malicious devices to a connected system as a motivation for using MFA for devices.

While still uncommon in the industry, applying two-factor authentication (a form of MFA) to IoT devices is target of recent research³⁰. The performance of the IoT network, and the suitability for lightweight IoT devices is referred to as challenge of applying MFA schemes and, therefore, in that work are presented studies to decrease hardware use in authentication schemes. This follows a similar motivation to the research line of the previously presented in ARCADIAN-IoT's network-based authentication (section 2.2).

The result of a successful MFA process should allow an entity to start using the functionalities and data of a given service. For this, in terms of the information that results of a MFA process, OpenID Connect defines ID tokens, which contain information on what happened on the authentication process³¹. The same source defines access tokens as what OAuth client uses to make requests to an API.

2.4.2.2 Research findings and achievements

The main research findings and achievements are the following:

- a. A vision and architecture for the MFA process for persons and devices, focused on achieving the project objectives and KPIs, well-accepted between the partners involved in the identity management of ARCADIAN-IoT (1GLOBAL, ATOS and UWS). This vision and architecture was co-created with IoT solution providers from the project consortium, to ensure a smooth integration in their services in WP5.
- b. A person authentication process joining simultaneously SSI (decentralized identifiers), Biometrics, and Network-based Credentials (stored at the SIM, in the hardware secure element).
- c. A IoT device authentication process joining Decentralized Identifiers (DIDs) and Network-based Credentials (stored at the SIM, in the hardware secure element).
- d. A process and technologies for integration with other ARCADIAN-IoT components, like Behaviour Monitoring, to allow the framework to respond to potential threats related with authentication with reduced human intervention.
- e. A process for authenticating persons with identity data captured using two devices, the personal device and a IoT device. The partners found that the MFA role in this process should be just for the person and for the device authentications. The remaining flow should be performed in an integration of the Biometrics component with the IoT solution provider (process described in the next sections).

2.4.2.3 Produced resources

The main produced resources are the architecture, the interface specification between the MFA and biometrics, SSI and network-based identification, and the prototypes of the MFA for person and device authentication, which are being integrated by IoT solution providers and demonstrated in the context of WP5.

³⁰ <https://link.springer.com/article/10.1007/s11227-021-04022-w>

³¹ <https://oauth.net/id-tokens-vs-access-tokens/>

2.4.3 Design specification

2.4.3.1 Logical architecture view

Error! Reference source not found. depicts a logical architecture of ARCADIAN-IoT Authentication component, an MFA technology. For simplification purposes, the IoT solution provider backend services are not represented.

Assuming the role of orchestrator, the MFA articulates the information of the 3 identification/authentication factors of the framework. Based on these components results it issues an ID token for the requesting entity authenticated operation.

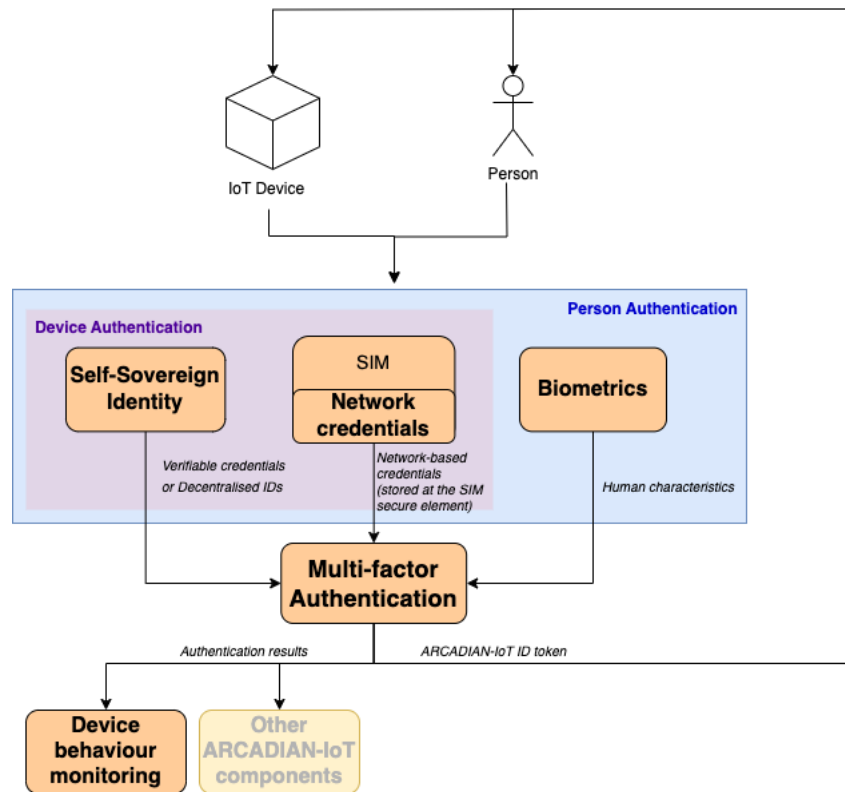


Figure 23 - ARCADIAN-IoT authentication high-level architecture

As outputs from the Authentication that feed other ARCADIAN-IoT components, there is the information provided to the Behaviour Monitoring, to allow this component to autonomously infer threats related with authentication requests (e.g. brute force attacks). After a successful authentication the MFA issues an ID token to be used for access management control / authorization according to the user-defined privacy rules, and for role base access control (defined by the IoT service providers). This can be done in ARCADIAN-IoT's Self-Aware Data Privacy component.

2.4.3.2 Sub-use cases

As sub-use cases of the Authentication component, there is the person authentication; the device authentication; and the person authentication with sources of authentication factors from 2 different devices.

2.4.3.2.1 Person authentication

According to the defined KPIs, person authentication should consider simultaneously 3 factors. The ones being considered are verifiable credentials, a network-based (hardware-based) identification (SIM/UICC-based³²); and biometrics.

2.4.3.2.2 Device authentication

Devices should have at least two robust identity mechanisms for devices. The MFA component will use for this purpose decentralized identifiers and a network-based (hardware-based) identification (SIM/UICC-based).

2.4.3.2.3 Person authentication with 2 different sources of information

This sub-use case considers the need of articulation of 2 different devices for person identification. An example of this case is Domain A's scenario where a drone needs to identify a person that requested its services, and the identification and authentication process needs to consider identity information from the personal device as well. As can be inferred, time and location will also be factors for identification in this process.

2.4.3.3 Interface description

In terms of interfaces, the MFA interacts mainly with the IoT service provider (IoT SP) and with the ARCADIAN-IoT identity management components. Authentication results are sent to a publish / subscribe broker for consumption by authenticated subscriber components like Behaviour Monitoring. Details about these interfaces can be found in Table 6.

Sender	Receiver	Communication type	Content exchanged
IoT SP	MFA	RESTful API	Authentication request with related identity claims
MFA	- SSI Authenticator - Biometrics Authenticator - Network Authenticator	RESTful API	Identity claims for verification of a given requester
- SSI Authenticator - Biometrics Authenticator - Network Authenticator	MFA	RESTful API	Results of identity claims verification
MFA	IoT SP	RESTful API	ID token (in case of authentication success)
MFA	- Behaviour Monitoring - Other authenticated ARCADIAN-IoT components	- RabbitMQ AMQP 0.9.1 for biometrics (authenticated publish/subscribe queue)	Authentication results for a given ARCADIAN-IoT ID

Table 6 - MFA interfaces

³² The technology developed is applicable to any SIM form (SIM, eSIM, iSIM)

For security purposes the MFA and the *authenticators* mentioned above can be deployed in each IoT solution provider premises, with the APIs only accessible internally by components that live in the same environment. This ensures the security of these components and ensures the sovereignty of each IoT solution provider identity material (of their devices and of their users). The technical specification and sequence diagram for these interfaces were agreed among the involved partners and is shared in the common project folder.

2.4.3.4 Technical solution

As previously mentioned, ARCADIAN-IoT authentication relies on multiple technologies, from multiple partners, to accomplish its objectives. This fact influenced the research strategy to achieve the technical solution, which, firstly, had the objective of settling a well-accepted vision for the component.

The achieved vision is depicted in [Error! Reference source not found.](#), where is defined the authentication flow for persons, joining the 3 identity factors defined as objective: the network identifiers stored at hardware level (SIM with subscriber identity), biometrics (facial characteristics) and self-sovereign identity (SSI) in the form of verifiable credentials (VC). The research done led to a definition of IoT devices authentication that is a sub-case of the persons authentication. This was intentional, to ensure a MFA component as holistic as possible, where the identification factors used can be modular / *plug and play* according to the needs. Therefore, the explanation in this section starts by depicting the persons' case, explaining after the differences for the devices' case.

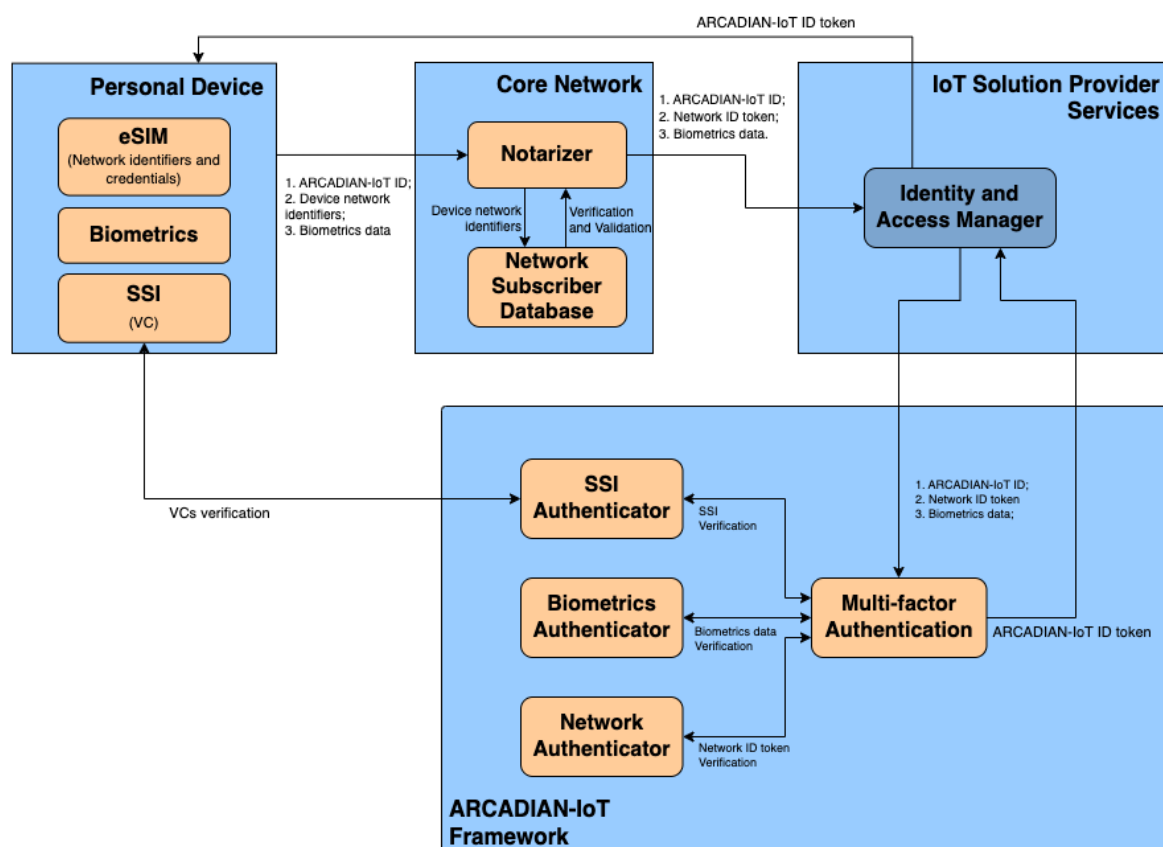


Figure 24 - Architecture from ARCADIAN-IoT MFA for persons

Before starting to analyse the flow in Figure 23, it is important to acknowledge that a pre-requisite for it to be possible is a successful registration process, where the person identification credentials were added to the framework, for being possible to be verified upon an authentication request. It

is assumed that after a successful registration the application running in the device (in the personal device in the case of a person authentication) has received an ARCADIAN-IoT ID, the framework unique identifier for that person. Now looking at the flow in Figure 23, for requesting the person authentication, the application should send the ARCADIAN-IoT ID to the IoT solution provider (SP), adding to it the biometrics data of the person needed for facial recognition. Implicitly, by communicating through cellular networks, it sends as well the network identifiers, which, passing through the network provider core infrastructure will be verified by a service called *Notarizer* which, in case of success, issues and signs a network ID token (for further details see the section on the network-based authentication). This network ID token is appended to the authentication request and forwarded to the IoT SP. The IoT SP forwards the request to ARCADIAN-IoT MFA, component that will:

- Request the SSI verification for that ARCADIAN-IoT ID. This implies the establishment of the communication with the SSI wallet running on the personal device from that person, for the Verifiable Credentials validation. For more details see the section on Verifiable Credentials.
- Request the biometrics verification, passing to a biometrics authenticator the biometrics data and the ARCADIAN-IoT ID. The authenticator will verify if biometrics received match the framework identifier. For more details see the section on Biometrics.
- Request the network authenticator to verify if the network ID token (a JWT) match the ARCADIAN-IoT ID received. This authenticator will verify the token validity (e.g. signature from the network operator and expiry time) and if its data matches the one expected for the framework identifier passed. For more details see the section on the Network-based Authentication.

In case of an unsuccessful verification of one of the credentials, the authentication fails and the IoT SP is informed of that. In the case of a successful verification of the 3 identity credentials from that ARCADIAN-IoT ID, the MFA will issue, encode and sign an ARCADIAN-IoT ID token following the common OIDC ID token structure, which will be returned to the IoT SP, and to its in the application personal device.

As previously mentioned, the IoT device authentication flow is very similar to the one described. The only change is that it doesn't use the biometrics credentials, as it would be expected, and that are used Decentralized Identifiers (DIDs) instead of Verifiable Credentials (VCs). These changes in the decentralized solution don't affect the MFA approach. Thus, to better understand the differences between VCs and DIDs, please see the related sections.

For the moment, and considering the authentication factors used, the multi-factor approach will consider that all the factors will be used simultaneously. This means that no step-up authentication is planned (see background for the definition of step-up authentication). For device authentication, step-up authentication could be relevant for the IoT network performance enhancement, and this can be considered as future work. For persons, the rationale behind verifying all factors simultaneously, is that the ones used don't request much interaction from the person. The hardware-based / network-based factor is zero-touch mechanism, which means that no interaction is needed (just autonomous operations from the personal device). The biometrics is based on an approach that is normal in apps nowadays (e.g. to use facial recognition to unlock an app). Considering this a common practice with very low user interaction, we considered that it doesn't harm the user experience. Therefore, there is only one factor that requests user interaction, which is the one related with the self-sovereign identity (SSI). For this reason, all the factors will be assessed in parallel, being the MFA component the element that requests claims verification, aggregates the results, and issues an ID token based on them, for the entity authenticated operation. This token doesn't define any access/authorization rule. It just informs if the authentication of a given ARCADIAN-IoT ID was successful or not.

After the successful authentication, when a person or device needs to request authenticated access to data or services from the IoT SP, it will have to present the token issued by the MFA. The authorization to access the data or services can be granted after the token verification

(following OIDC standard procedures). Furthermore, ARCADIAN-IoT's Self-aware Data Privacy can be applied to manage the authorization rules based on privacy-related definitions or business rules specified by the IoT SP.

Person identification with 2 devices

Regarding the person authentication with simultaneous identification from 2 devices, the identity management partners (1GLOBAL, ATOS, and UWS), jointly with the Domain Owner where the technology will be applied in ARCADIAN-IoT (Domain A, LOAD – with the IoT solution on vigilance with drones in smart cities), agreed in the following solution:

- a. The person to be identified is already authenticated using ARCADIAN-IoT MFA above mentioned, using for such its personal device.
- b. The IoT device that will identify the person has also successfully authenticated using ARCADIAN-IoT MFA authentication.
- c. In the IoT application of the person's device, the person requests the identification service by the IoT device – e.g., in ARCADIAN-IoT Domain A, the person requests the drone vigilance service, which implies that the drone will need to identify the person.
- d. In a scenario of multiple IoT devices that can be used for the person identification, the IoT solution provider attributes the service to one particular IoT device (already authenticated).
- e. Considering that the person identification done by the IoT device will use biometrics, specifically facial recognition, both the person and the IoT device need to be in the same location at the same time.

Considering the conditions above, the person that was firstly identified with credentials gathered from her personal device, can now be identified by a second (IoT) device, to whom the service was attributed to, which is able to gather biometrics data to recognize her, and is in the same place and time of the person. The IoT SP, having in place all the factors (both entities authenticated, the service data, and the time and location of both) and requesting the drone captured biometrics data verification from the Biometrics component, has a very strong identification mechanism. In the case that all conditions are met, the service may proceed. As described, in this case, the MFA component is only used for authentication of the persons and devices involved, being the remaining flow managed by the IoT SP with the support of the Biometrics component.

2.4.4 Evaluation and results

The ARCADIAN-IoT MFA solution has been iteratively deployed, integrated, and assessed in several use cases provided by the consortium, particularly the ones related with the personal surveillance with drones, and medical IoT. This ensures that the final solution is adjusted to the IoT solution providers' needs, while ensuring strong authentication for the entities participating in their IoT ecosystems – persons and IoT devices. Particularly in medical IoT context the agnostic nature of the solution has been verified, being used for several types of actors – e.g., the patient authentication through the medical kit provided, which included a smartphone, and for the medical staff authentication as well. Final use cases integration and demonstration of this technology is ongoing in ARCADIAN-IoT WP5 in the 2 different IoT domains mentioned.

The solution is also integrated with other ARCADIAN-IoT components, namely with the Behaviour Monitoring, which will infer threats based on authentication attempts. Furthermore, the authenticated operation of the IoT entities is ongoing as expected, with the ARCADIAN-IoT token validation according to OIDC standard process.

Recalling ARCADIAN-IoT objectives that related with this component, these were of *enabling security and trust in the management of objects and persons' identification*, where the component should support at least 2 robust identity mechanisms for devices; and at least 3 multiple simultaneous identification approaches for persons. Considering that ARCADIAN-IoT MFA orchestrates, for persons, simultaneously, the hardware/network-based identification, the decentralized identification, and the biometrics identification, and that for devices, it orchestrates

the same identifier types except biometrics, it is possible to determine that it contributed to the successfully achievement of the objectives.

The research done in the scope of this component also led to a solution for the person identification with 2 devices, previously described, which is being demonstrated in the scope of WP5.

2.4.5 Future work

The immediate future work for ARCADIAN-IoT MFA, is its demonstration and dissemination of its application in the IoT solutions targeted in WP5. The final definition regarding its exploitation path is also in progress in the next few months, being the defined exploitation actions put in place after the project.

3 TRUST PLANE

3.1 Verifiable Credentials

3.1.1 Overview

3.1.1.1 Description

ARCADIAN-IoT will provide an identity management solution that is built on W3C Verifiable Credentials specification [7] that is a core standard for the Self-Sovereign Identity (SSI) approach. The solution enables trusted identification of users and things through the issuing of identity claims as Verifiable Credentials (VCs) to their respective secure crypto based digital identity wallets and agents without depending on centralised Identity Providers with its inherent privacy risks. Once a user or thing has been issued with Verifiable Credentials, they can later present them to other entities such as services and apps which require to authenticate the user or thing in a trusted crypto based manner, that only the holder of the requested Verifiable Credential can do.

Decentralised Identifiers described in section 2.1 provide an identity that is resolvable over a decentralised and distributed infrastructure to cryptographic keys associated to the identity. This helps provide for the digital signature validation of issued Verifiable Credentials by the issuer and the presentation of Verifiable Credentials to 3rd parties, which combine to underpin the root trust in Self-Sovereign Identity.

Verifiable Credentials are supported by an SSI identity framework that is discussed in section 3.1.2 to provide the core building blocks for issuing, presenting and verifying credentials as per the W3C Verifiable Credentials specification.

3.1.1.2 Requirements

A recall of the requirement 5.1.1 first defined in D2.4 is given below and it is also supplemented with additional related sub- requirement.

- **Requirement 5.1.1 – Verifiable Credential management**
 - To provide Verifiable Credential based identity management to enable secure and authenticated identity and other claims needed by the services and apps in the IoT ecosystems

3.1.1.3 Objectives and KPIs

The overarching objective is to employ the Verifiable Credentials protocol for integration in the Permissioned Blockchain and implement / support the integration of the different agents (issuer, holder and verifier, as defined by the W3C VC specification) for the developed components and use cases.

Additional aims are as follows:

- Verifiable Credentials will allow any system or user to cryptographically verify in real time claims related to the IoT device.
- Further enhance trust through VCs by enhancing implementations towards standard's interoperability.
- Use Verifiable Credentials in combination with Decentralized Identifiers (DIDs), with trust rooted on Distributed Ledger Technology (DLT), to ensure the authenticity, integrity, immutability, and uniqueness of each object without relying on a Central Authority.
- A desirable objective is to support eIDAS Bridge [20] within the ESSIF project where a service can issue Verifiable Credentials to a user.

KPIs defined for Verifiable Credentials are listed below:

KPI scope	
Support at least one domain use case with Verifiable Credentials	
Measurable Indicator	
Number of domains using Verifiable Credentials	
Target value	Achieved value
1	2

KPI scope	
Interoperability with at least one eIDAS identity schema.	
Measurable Indicator	
Issue person Verifiable Credential with an eIDAS compatible schema.	
Target value	Achieved value
1	1

KPI scope	
Enable, at least 3 multiple simultaneous identification approaches for persons.	
Measurable Indicator	
Support Verifiable Credential identification from a person's mobile wallet.	
Target value	Achieved value
1	1

KPI scope	
Support, at least two robust identity mechanisms for devices and apps/services.	
Measurable Indicator	
Devices and apps/services support SSI mechanisms in at least in one domain.	
Target value	Achieved value
2	2

3.1.2 Technology research

In this section it is described the background that ATOS brings to ARCADIAN-IoT in Self-Sovereign Identity and proceeds to examine the State-Of-The Art considering the scope of standardisation in this new technology area and the need for interoperability before doing a final technical analysis on the competing technology to appraise the selected technology for ARCADIAN-IoT.

3.1.2.1 Background

During the first version of this deliverable D4.1 [46], ATOS was in the process of building a Self-Sovereign Identity prototype called Ledger uSelf, with the aim to simplify the adoption and integration of Self-Sovereign Identity by Service Providers. The prototype provided a mobile wallet and also a broker that acts as a wrapper on top of an open source Hyperledger Aries GO Agent [42], which in turn makes integration easier for Service Providers. The Ledger uSelf solution had basic support for issuing, presenting and verification of Verifiable Credentials for persons. The figure below shows the overall Aries SSI Agent design with all features implemented as per the Aries Protocol Request for Comments (RFCs)³³ and shows the external interfaces supported by the Hyperledger Aries SSI Agent towards other Aries SSI Agents (including deployed in SSI Wallets and Mediators) as well as the Ledger uSelf Broker and its interface to the Relying Party.

³³ <https://github.com/hyperledger/aries-rfcs>

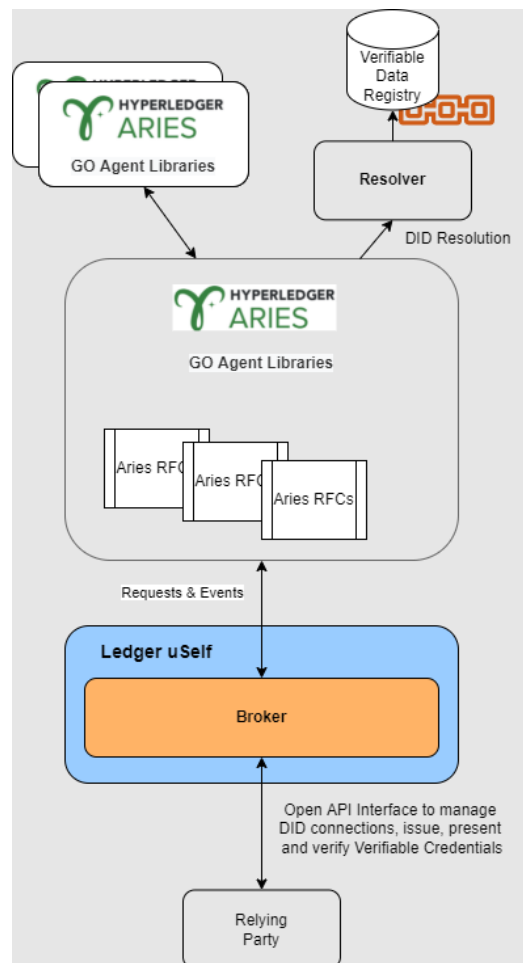


Figure 25 - Ledger uSelf built on top of Hyperledger Aries GO Agent

The prototype Ledger uSelf Broker prototype was implemented in Kotlin.

The following sub-sections investigate the current state-of-the-art in this field, and also the work being done on interoperability to make sure that implementations can fully interwork with each other.

3.1.2.1.1 State of the Art

There are several Self-Sovereign Identity solutions on the market today based on the evolving standards of Decentralized Identities and Verifiable Credentials. As input to the technology choice for ARCADIAN-IoT a range of solutions available at the time of this analysis³⁴ is considered as follows:

- **Veramo [23]** is an evolution of the uPort open source SSI software that was one of the first pioneers of SSI from 2015. uPort's technical architecture and open source libraries started to manifest limitations due to changes in maturing standards that meant many changes throughout the code base and also its tight integration with on-chain identities. An evolution to a new open source framework has resulted in a new modular architecture based around a library of core functionality, which allows the developer community to easily interface with and extend its functionality as needed, for example with additional DID methods, key management, protocols.

³⁴ The initial analysis in the first deliverable D3.1 was carried out in the first half of 2022. This was not revisited in subsequent deliverables as the technology choice was made at this time, however it should be noted that it is a fast-evolving field and the SOTA landscape has changed significantly.

- **Veres One [24]** is a non-profit identity project with the goal of addressing a range of existing identity challenges. Veres One supports a public permissionless network and the cost to create a Decentralized Identifier is approximately one US Dollar.
- **Hyperledger Indy [25]** originally developed by Evernym [141], is a public permissioned DLT solution purpose-built for decentralized identity and initially integrated with the Sovrin Blockchain Network.
- **Hyperledger Aries [26]** is an open source evolution from Hyperledger Indy that creates a modular and extensible SSI Framework that is completely independent of any Verifiable Data Registry, be it based on DLT or otherwise. Aries has notably led standards-based interfaces through its work in W3C and the Decentralised Identity Foundation (DIF).
- **Jolocom [27]** is another open source SSI platform that uses Ethereum by default. It uses hierarchical deterministic keys to create multiple identities from a seed master identity and resultant DIDs resolve to a DID Document stored on IPFS.
- **MATTR [28]** have developed an open and standards-based decentralized identity platform. They have a strong identity product and also open-source software including a mobile wallet built on Hyperledger Aries. They provide SSI solutions as well as providing configurable building blocks to suit a broad array of use cases and user experiences.
- **SpruceID [29]** builds open-source credentialing infrastructure that is standards-compliant, production-ready, and extensible into typical enterprise and government IT systems. SpruceID SSI provides open-source and standards-based core Verifiable Credential and Decentralized Identifier functionality in Rust.
- **IOTA Identity [30]** is an open-source and standards-based Rust implementation of decentralized digital identity. It implements standards such as the W3C Decentralized Identifiers (DID) and Verifiable Credentials and the DIF [DIDCOMM Messaging](#). This framework can be used to create and authenticate digital identities, creating a trusted connection and sharing verifiable information, establishing trust in the digital world. It is integrated and tested with the IOTA Tangle DID method as described in section 2.1 although the components themselves are ledger agnostic. Current version is 0.5.0 and the notice reads: *“This library is currently in its **beta** stage and under development and might undergo large changes! As such, it is to be seen as experimental and not ready for real-world applications”*.
- **AlastriaID [31]** is deployed as one of the basic applications of the promoted blockchain infrastructure by the Alastria consortium within its platform. This technological digital identity in blockchain aims to provide and establish an infrastructure and development framework, to carry out Sovereign Digital Identity projects, with full legal force in the euro zone. The implementation design follows W3C standards with some important differences in their blockchain based DID and VC specifications and the VC token design and use of hashes.

3.1.2.1.2 Interoperability

Overview

Due to initial SSI developments preceding much of the standards work and differing rival technologies, their implementations were never going to be able to interwork with each other. However, today there is a lot of effort going into interoperability as the standards have matured. There are, however, still many challenges aside from doing interoperability tests to make different interpretations of the standards interwork with each other. As we can see in the following table from the Decentralized Identity Foundation Interoperability WG [32] there are rival protocols supporting different SSI stack approaches for the VC data model, exchange, proof presentations and transport.






	Stack				
Layer	WACI-PEX 	OIDC SIOPv2 	Aries Proposed 	Aries AIP 2.0 	Aries AIP 1.0 
Data Model	Verifiable Credentials		AnonCreds and Verifiable Credentials		AnonCreds
Exchange	PEX		PEX and AnonCreds		AnonCreds
API	WACI	OIDC4VP + Claims Aggregation	Present-proof-v3	Present-proof-v2	Present-proof-v1
Communication/Transport	DIDComm V2 + transports	OIDC + SIOP	DIDComm V2 + transports	DIDComm V1 + DIDComm V2 Envelope + transports	DIDCOM V1 + transports

Figure 26 - Protocol support for SSI [32]

Additionally, not all implementations support the same cryptographic identity, signatures and proofs, as we see below.

Layer	Verifiable Credential Technology		
Cryptographic Identity	DID	Key	Linked Secret
Signature/Proof	ES256, ES256K, EdDSA, Ed25519SignatureXyz, BBS+, etc.		
Credential	JSON-LD	JWT	X.509
Presentation	JSON-LD	JWT	

Figure 27 - Cryptographic technology [32]

Hyperledger Aries, MATTR, Spruce and Veramo are amongst the most active participants in this Interoperability WG. Interoperability testing is also supported by W3C with issuing and verification of VCs for testing available here³⁵.

DID Exchange and VC Presentation

For SSI agents to be able to provide applications with a secure, private communication methodology they are built on top of decentralized design making use of DIDs (see section 2.1). This enables agents to reliably exchange DIDs and verify each other as the holder of that DID and reliably share Verifiable Credentials, all with cryptographic proofs based on the DIDs. Currently there are two rival protocols to perform this:

1. DIDCOMM is a dedicated Self-Sovereign Identity standard-based protocol that arose from Aries (now standardised in DIF) and is needed to be supported by devices and services alike so that they can successfully interwork with each other.
2. Self-Issued OpenID Provider v2 (SIOP) [35] and OIDC-4-Verifiable-Presentations (OIDC4VP) [34] build upon the well established OIDC protocol, but now with the OpenID Provider under the End-User's local control. End-Users can leverage Self-Issued OPs to authenticate themselves and present claims directly to Relying Parties (RPs).

With two rival protocols there is dilemma in which one to support. DIDCOMM comes from Aries, builds on standard based JWM [36] and is now standardised in DIF, whereas SIOP / OIDC4VP have been developed more recently and build on OIDC so that it makes use of technology that is already well supported and understood by many online services and identity providers. So, taking the OIDC approach would help with one of the major challenges of using new technology, that being adoption.

³⁵ <https://github.com/w3c-ccg/vc-api>

However, for now, it seems that any SSI solution would need to be able to support both DIDCOMM and Self-Issued OpenID Provider protocols to be interoperable with the varying SSI solutions.

EBSI ESSIF Interoperability Profile

EBSI ESSIF have created an Interoperability profile [33] to make sure that the infrastructure they are creating for issuing Verifiable Credentials in the ecosystem will be able to work with many different implementations which is very much needed if SSI is to be adopted ubiquitously. It is also observed that EBSI are currently only promoting SIOP / OIDC4VP interwork. However, other important frameworks promoted by the European Commission such as IOTA and GAIA-X also use DIDCOMM.

Technical analysis

SpruceID and IOTA both build in RUST, for its suitability across many different platforms including embedded systems due to its memory safety, amongst other features. This makes them more suitable for applications in constrained IoT Devices.

Currently, however IOTA Identity seems to be somewhat early to adopt as it is still in Beta and to date also only integrated with IOTA Tangle DID - which we already seen was not in line with the GDPR “right to be forgotten” principle and is not known to be active in interoperability efforts. SpruceID on the other hand is seen to support a comprehensive open-source solution and is active in interoperability efforts as can be seen here [37] and also with their participation in the Decentralized Identity Foundation Interoperability WG.

Hyperledger Aries is a fully open-source framework that has a strong development team with continued releases and pushing the standards to promote the interoperability of SSI. It is a state-of-the-art dedicated framework supporting SSI Agents (available in Python, .NET, GO) that implement the core features and offer APIs to be integrated with 3rd party applications. It commenced in 2019 and is now quite mature with good documentation, supports interoperability and testing, and has its latest release coming out in April, so to keep in check with the latest updates in the standards as shown here³⁶. It is amongst these characteristics that ATOS Research and Innovation chose Hyperledger Aries GO to build its Self-Sovereign Identity solution called Ledger uSelf.

In summary, SpruceID and Hyperledger Aries GO are primary candidate open-source technologies for consideration in developing SSI for persons and things according to the analysis carried out here. That said, SpruceID is a small start-up enterprise whereas Hyperledger Aries GO is part of the Hyperledger Foundation global collaboration, hosted by The Linux Foundation and has a larger coding community and also more active as can be seen by its GitHub Repo insights [43]. Moreover, hyperledger Aries is seen to benefit from a wider collaboration and has extensive and continued development of its standard SSI framework.

Considering the above and that ATOS’s own prototype implementation, discussed in section 3.1.2.1, currently provides for basic Verifiable Credential issuing, presentation and validation, it is decided to further develop the ATOS Ledger uSelf asset (based on Hyperledger Aries GO) to meet the needs of ARCADIAN-IoT with the primary objectives to add support for IoT Devices and privacy preserving ZKP in presenting Verifiable Credentials (see section 2.1.2.1.5).

3.1.2.2 Research findings and achievements

Current state-of-the-art analysis in Self-Sovereign Identity and support of Verifiable Credentials was carried out and the analysis concluded that Hyperledger Aries was the best choice to support Verifiable Credentials in the ARCADIAN-IoT framework, considering also that it has been adopted by GAIA-X and IOTA³⁷.

³⁶ <https://github.com/hyperledger/aries-framework-go/releases>

³⁷ Note: Hyperledger Aries was the SSI framework adopted by GAIA-X and IOTA at the time of the first version of this deliverable in April 2022 which was the basis of the technology choice at the time, however both have since moved away from Hyperledger Aries and adopted the OpenID SSI standards.

3.1.2.3 Produced resources

In addition to enhancing the ledger uSelf SSI solution for ARCADIAN-IoT, as described in the following sections, a new SSI IdP component is implemented for registering ARCADIAN-IoT entities (persons, IoT-Devices, services) to the ARCADIAN-IoT Framework and handling frontend and backend interactions with the Ledger uSelf Broker for issuing and verifying Verifiable Credentials.

3.1.3 Design specification

3.1.3.1 Sub-use cases

3.1.3.1.1 *Issue a Verifiable Credential to a Person's SSI Wallet*

A person is able to be issued with a Verifiable Credential to their SSI Wallet which is later able to be used for onboarding in ARCADIAN-IoT framework and Service Provider services.

3.1.3.1.2 *Issue a Verifiable Credential to an Organization Member's SSI Wallet*

A member of an organization can be issued with a Verifiable Credential to their SSI Wallet which is later able to be used for onboarding in ARCADIAN-IoT framework and Service Provider services. This is based on the organization's records provisioned with the person eID and requesting to first verify the user's Person VC before the user is issued with an Organization Member VC.

3.1.3.1.3 *Issue a Verifiable Credential to an IoT Device's SSI Agent*

An IoT Device can be issued with a Verifiable Credential to their SSI Agent which is later able to be used for onboarding in ARCADIAN-IoT framework and Service Provider services.

3.1.3.1.4 *Present a Verifiable Credential from a Person's SSI Wallet*

A person is able to present a Verifiable Credential from their SSI Wallet so to prove their identity credential.

3.1.3.1.5 *Present a Verifiable Credential from an Organization Member's SSI Wallet*

An Organization Member can present a Verifiable Credential from their SSI Wallet so to prove their identity credential.

3.1.3.1.6 *Present a Verifiable Credential from an IoT Device's SSI Agent*

An IoT Device is able to present a Verifiable Credential from its SSI Agent so to prove its identity credential.

3.1.3.1.7 *Verify a Verifiable Credential received from a Person's SSI Wallet*

The ARCADIAN-IoT framework implements an SSI Agent so to verify a person's Verifiable Credential.

3.1.3.1.8 *Verify a Verifiable Credential received from an Organization Member's SSI Wallet*

The ARCADIAN-IoT framework implements an SSI Agent so to verify an Organization member's Verifiable Credential.

3.1.3.1.9 Verify a Verifiable Credential received from an IoT Device's SSI Agent

The ARCADIAN-IoT framework implements an SSI Agent so to verify an IoT Devices Verifiable Credential.

3.1.3.1.10 Authenticate an IoT Device's SSI Agent with its PUBLIC DID

Where a constrained IoT Device is not able to support the full SSI stack due to resource limitations it should at least be verified by its controlling IoT GW proving that it is in possession of the DIDs private key, by performing DID Authentication as described in 2.1.3.4.3.

3.1.3.1.11 Service Provider a Person in the ARCADIAN-IoT framework

Persons are onboarded by a Service Provider service to the ARCADIAN-IoT framework. The use case figure below shows the onboarding process for an end user in a service provided by the ARCADIAN-IoT register person. This also shows the related pre-requisite sub-use case of "Issue a Verifiable Credential to a Person's SSI Wallet".

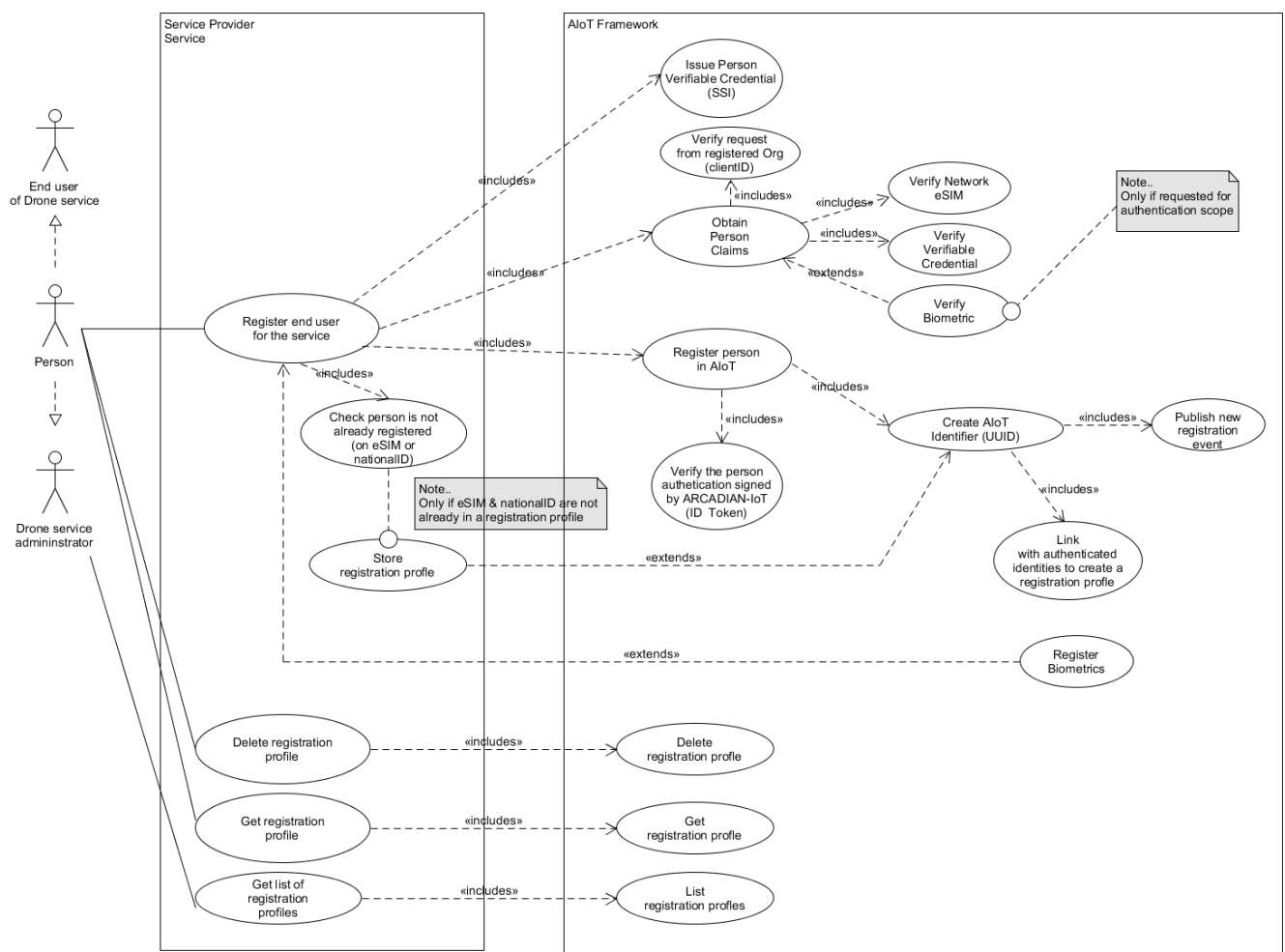


Figure 28 - Register Person in ARCADIAN-IoT Framework by a SP service

Note that it is only authorised for Service Provider services registered in the ARCADIAN-IoT framework to be able to perform person onboarding / registration (see section 2.1.3.1.6).

3.1.3.1.12 Service Provider registers an Organization member in the ARCADIAN-IoT framework

Organization Members are onboarded by a Service Provider to the ARCADIAN-IoT framework.

3.1.3.1.13 Service Provider registers an IoT-Device in the ARCADIAN-IoT framework

IoT-devices are onboarded by a Service Provider to the ARCADIAN-IoT framework.

3.1.3.1.14 Service Provider updates a registered Person, Organization Member or IoT-Device identity in the ARCADIAN-IoT framework

Previously onboarded Persons and IoT devices are updated by a Service Provider in the ARCADIAN-IoT framework.

It is only authorised for Service Provider services registered in the ARCADIAN-IoT framework to be able to perform onboarding (see section 2.1.3.1.6).

3.1.3.1.15 Service Provider deletes a registered person, Organization Member or IoT-Device in the ARCADIAN-IoT framework

A Service Provider can delete a person or IoT Device that they have previously registered in the ARCADIAN-IoT framework.

3.1.3.2 Logical architecture view

The following figure shows the logical architecture of the Ledger uSelf Self-Sovereign Identity solution with the broker supporting easy integration of Hyperledger Aries Agent and with capabilities to extend the functional capabilities.

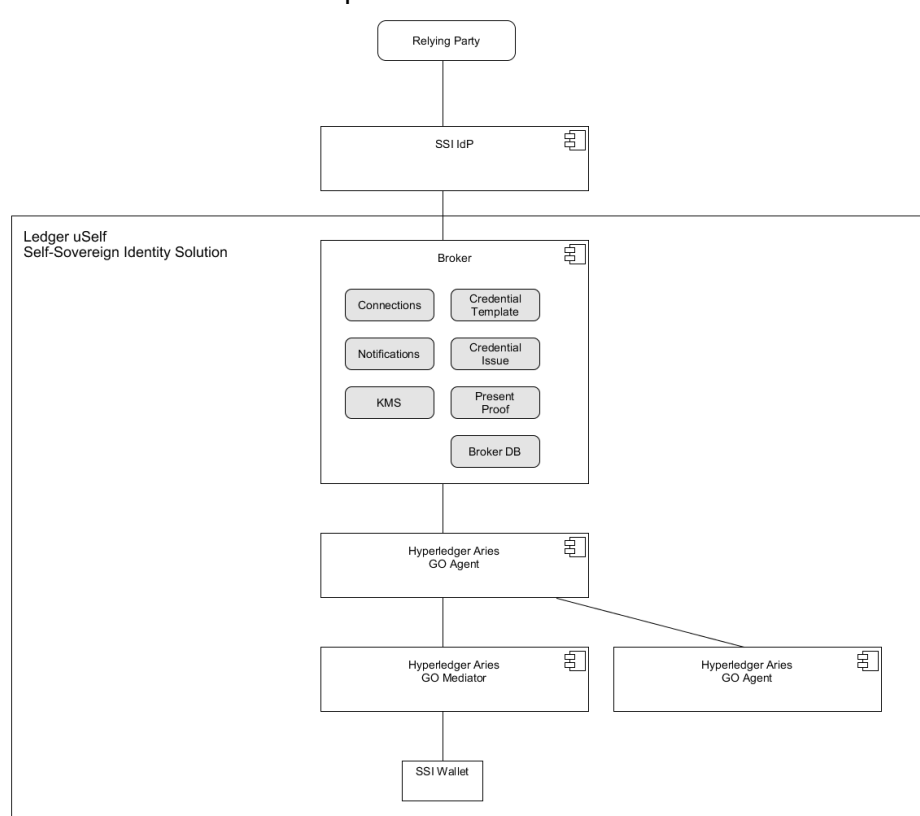


Figure 29 - Ledger uSelf Broker Self-Sovereign Identity Solution + SSI IdP

The above figure shows the Broker as a distinct component from the Hyperledger Aries GO Agent for the P1 deployments. Note that for P2 it will be one integrated SSI Agent / Broker in GO so to be able to operate more efficiently on more restricted IoT Devices such as the Jetson Nano in Domain A.

The Ledger uSelf Self-Sovereign Identity solution consists of the following components:

Broker: This component is developed in Kotlin and acts as a wrapper over the Hyperledger Aries Agent to simplify and make easier for organisations to integrate a Self-Sovereign Identity solution as another authentication means. Typically, it integrates between Relying Parties and the Hyperledger Aries GO Agent.

Hyperledger Aries GO Agent: This is the open-source Hyperledger Aries Agent developed in GO [42] which supports the base SSI functions for Issuing, Presenting and Verifying Verifiable Credentials as in line with the W3C Verifiable Credential specification [7]. The Aries GO Agent interacts with other agents or SSI Wallets via the mediator over the DIDCOMM protocol [22].

SSI Wallet: The mobile wallet application implements an SDK that integrates to the Aries GO Agent converted to run on android and provides a user interface for being issued with and presenting Verifiable Credentials.

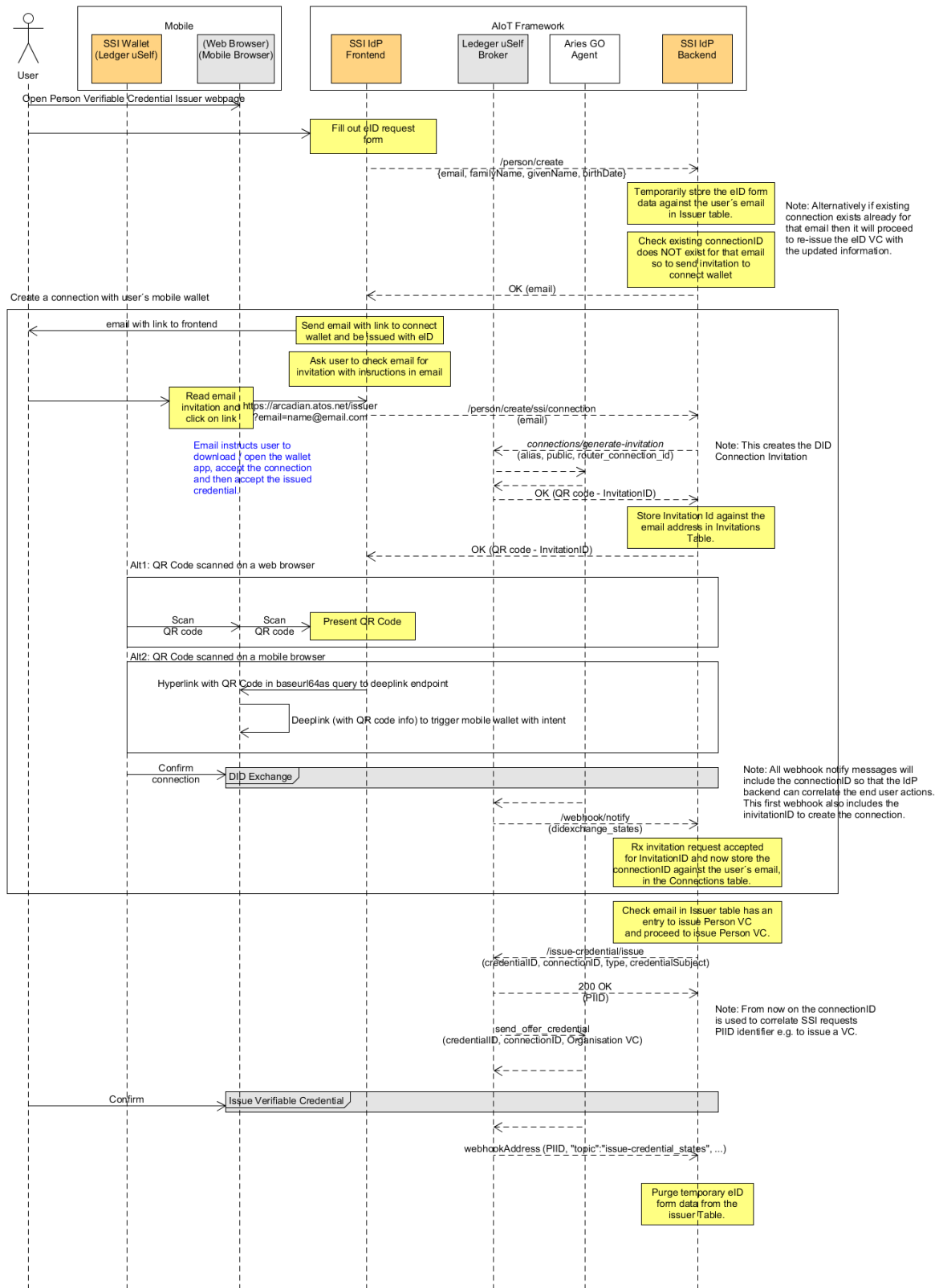
Hyperledger Aries GO Mediator: This is a specific instance of the Hyperledger Aries Agent implemented to handle full-duplex communications with mobile devices using websockets. This provides a standardized way for the GO Mediator to send content to the SSI Wallet without being first requested by the SSI Wallet and therefore enables messages to be passed back and forth while keeping the connection open.

In addition to the above Ledger uSelf solution components it can be seen in the above figure that ATOS provides also another component:

SSI IdP: This component provides several functionalities regarding the identity provision of persons and devices in ARCADIAN-IoT. Firstly, it provides a Verifiable Credential issuer role so to support the issuing of persons and devices with Verifiable Credentials as a pre-requisite for all of the ARCADIAN-IoT use cases. It also supports a front-end for the requesting of Verifiable Credentials and for authentication use cases where a user is requested to present a Verifiable Credential. Finally, it supports the onboarding of persons and devices to Relying Parties and also in the framework, with an ARCADIAN-IoT Identity (aiotID) generated and published to all ARCADIAN-IoT component services that subscribe to the aiotID registration event.

3.1.3.3 Sequence diagrams

3.1.3.3.1 Issue a Verifiable Credential to a Person's SSI Wallet (P1)



3.1.3.3.2 Issue a Verifiable Credential to an Organization Member's SSI Wallet (P2)

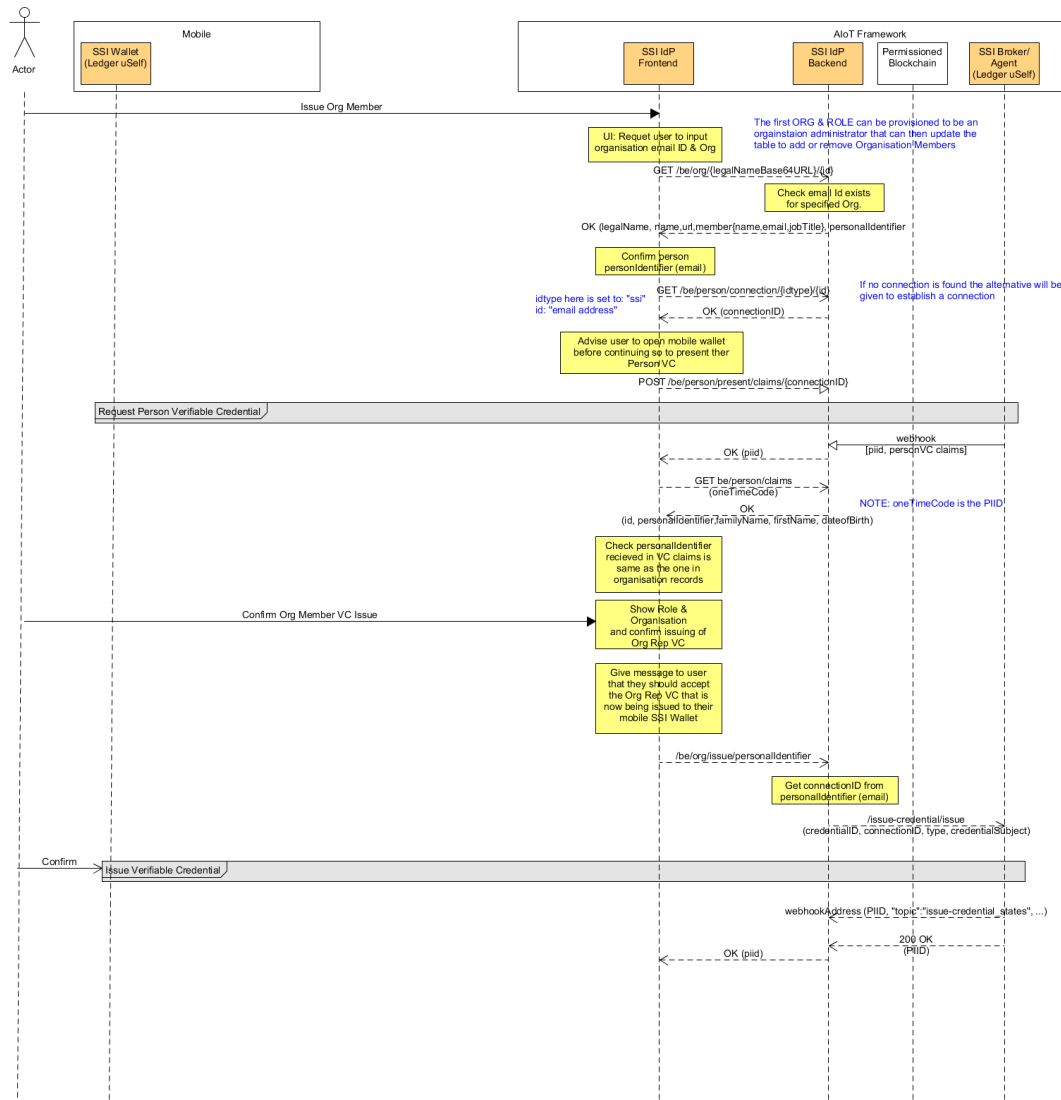


Figure 31 Issue an Organization Member VC

3.1.3.3.3 Issue a Verifiable Credential to an IoT Device's SSI Agent (P2)

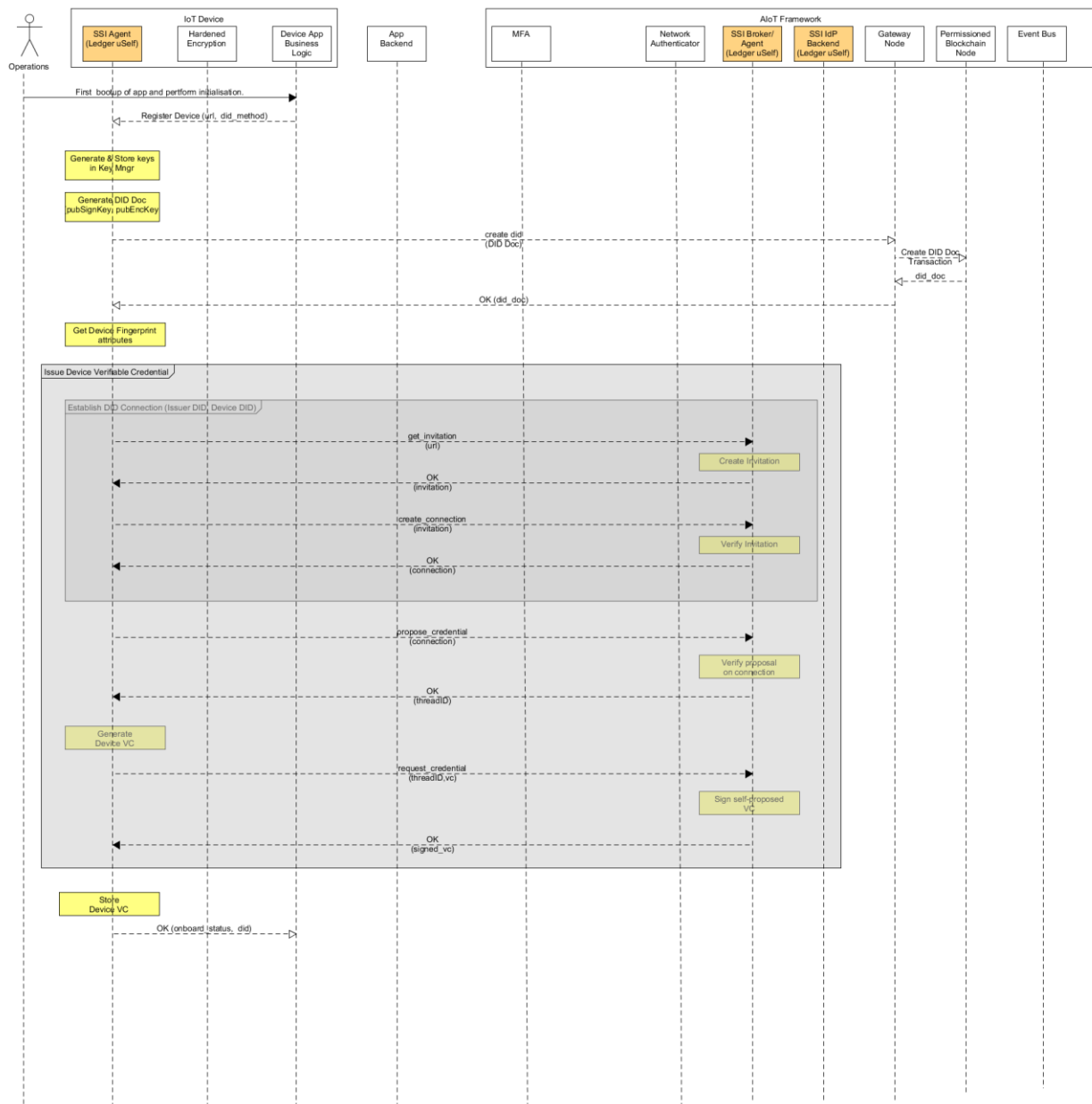


Figure 32 Figure 30 - Issue a Device VC

3.1.3.3.4 Present a Verifiable Credential from a Person's SSI Wallet (P1)

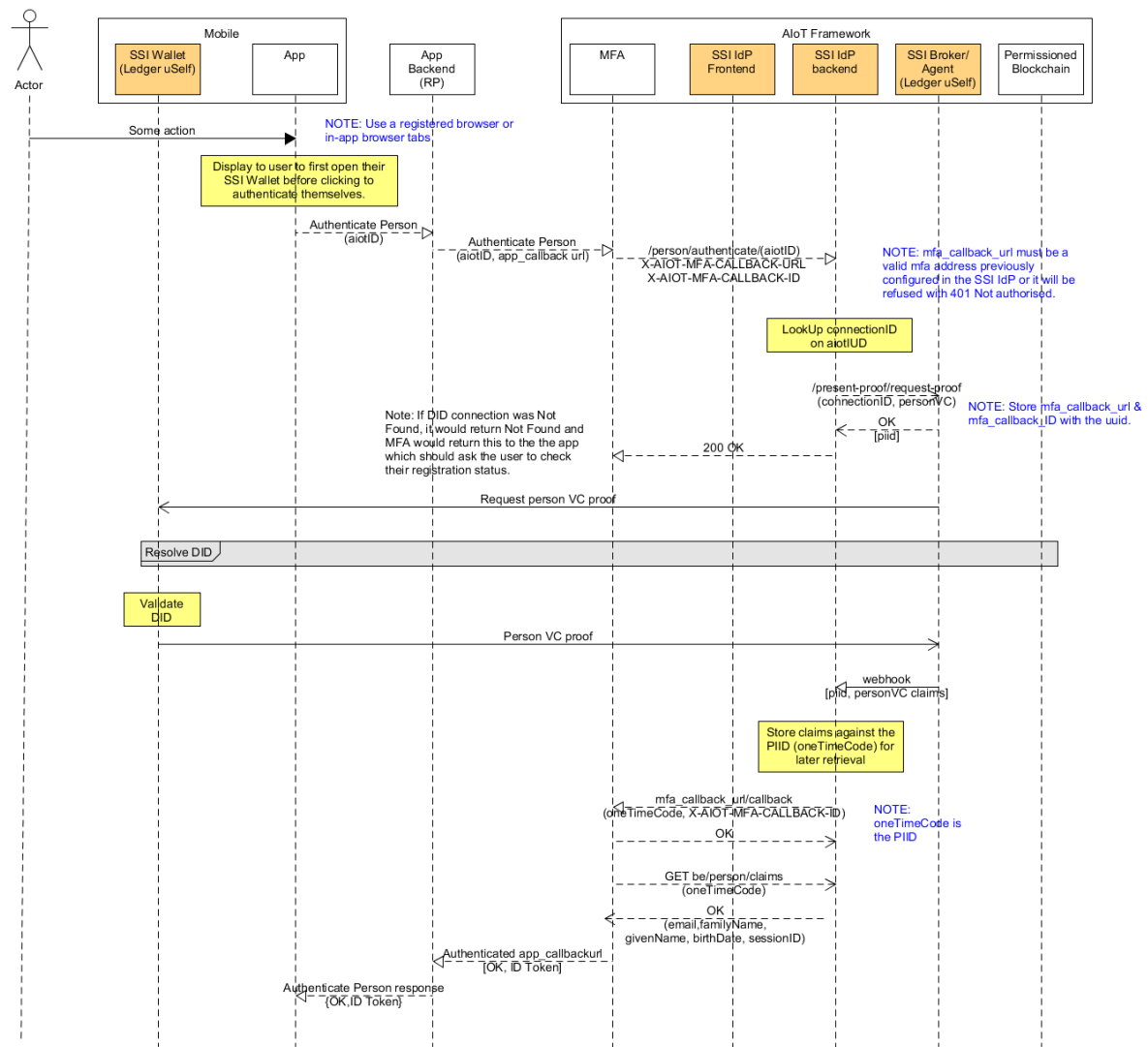


Figure 33 – Present and verify a Person VC

Note that if the aiotID was not presented by the service, the SSI IdP would query the user for the identity associated with their identity Verifiable Credential. For a Person VC this will be the user's email address. In the case that national eID were supported this would be their national identity.

3.1.3.3.5 Present a Verifiable Credential from an Organization Member's SSI Wallet (P2)

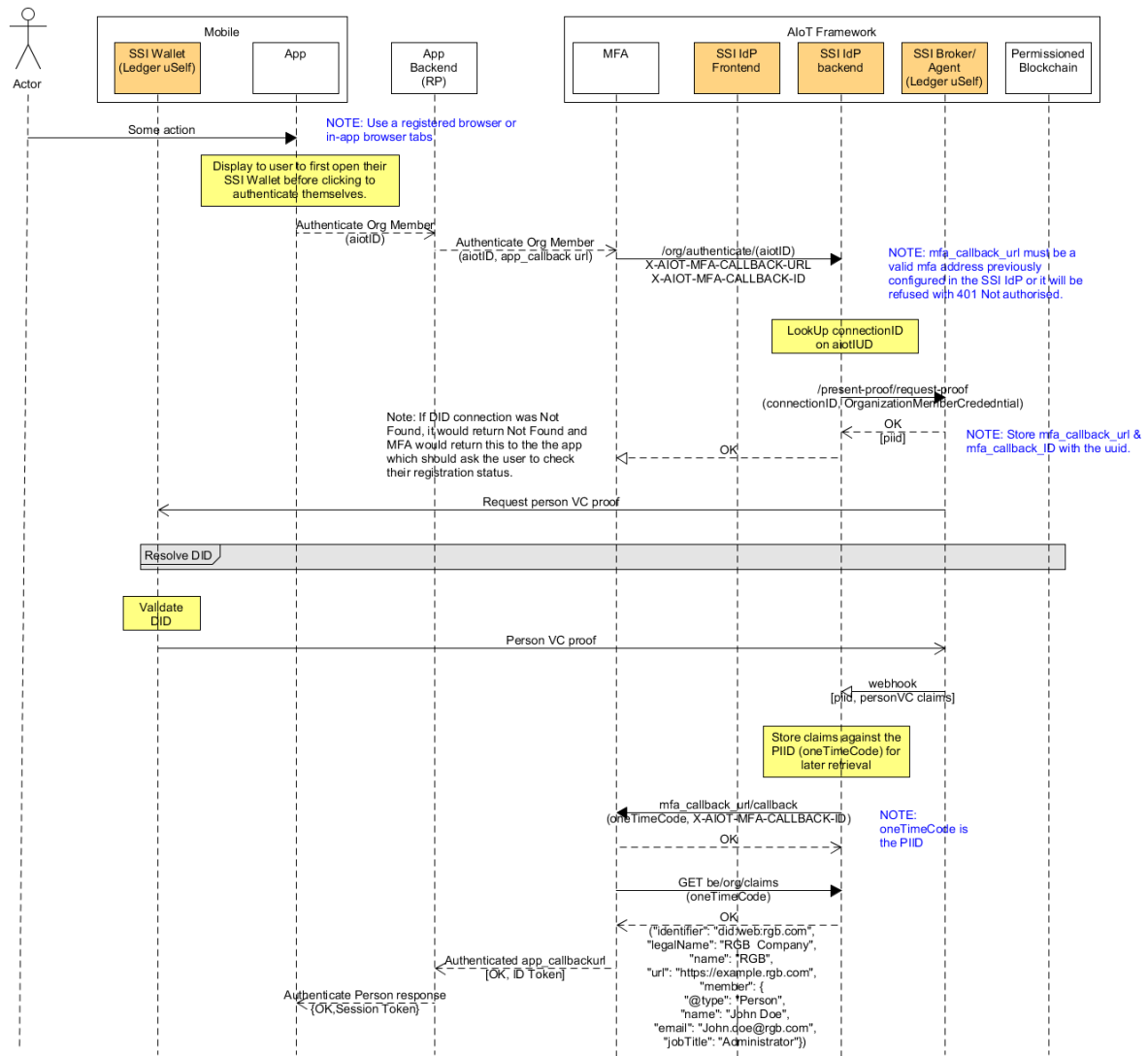


Figure 34 – Present and verify an Organization Members VC

3.1.3.3.6 Present a Verifiable Credential from an IoT Device's SSI Agent (P2)

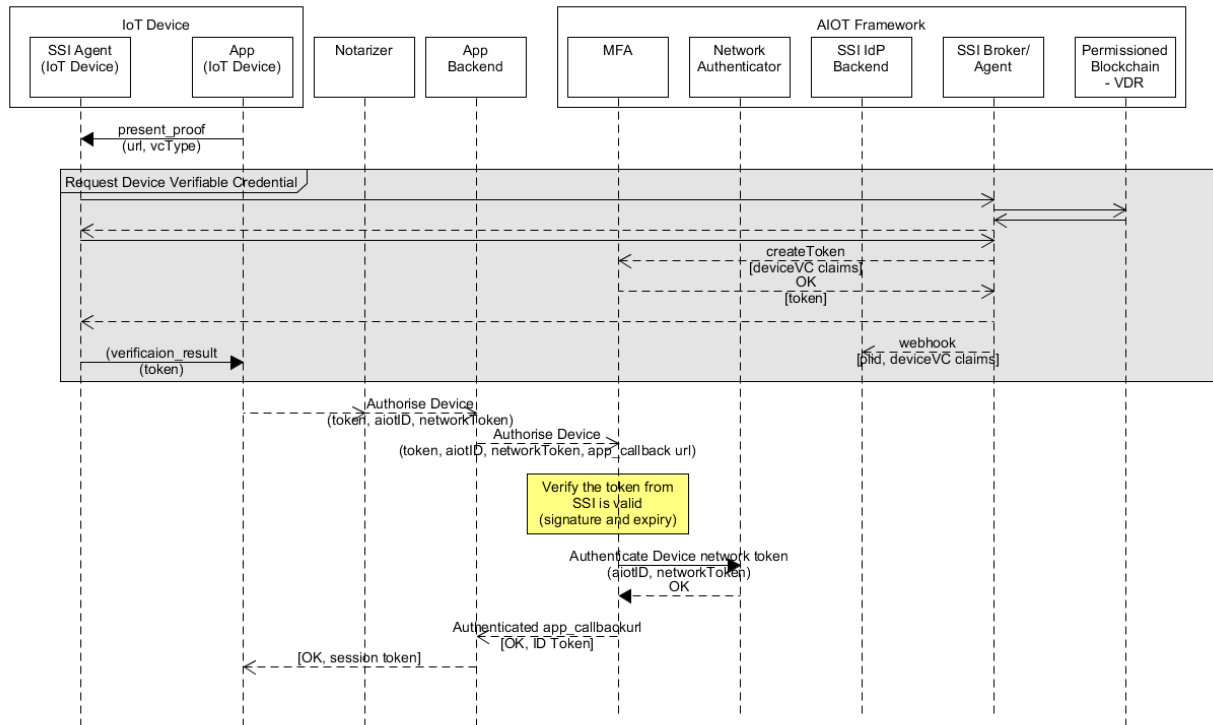


Figure 35 Present and verify a Device VC

3.1.3.3.7 Verify a Verifiable Credential received from a Person's SSI Wallet (P1)

See section 3.1.3.1.4.

3.1.3.3.8 Verify a Verifiable Credential from an Organization Member's SSI Wallet (P2)

See section 3.1.3.3.5

3.1.3.3.9 Verify a Verifiable Credential received from an IoT Device's SSI Agent (P2)

See section 3.1.3.3.6.

3.1.3.3.10 Authenticate a constrained IoT Device's with its PUBLIC DID (P2)

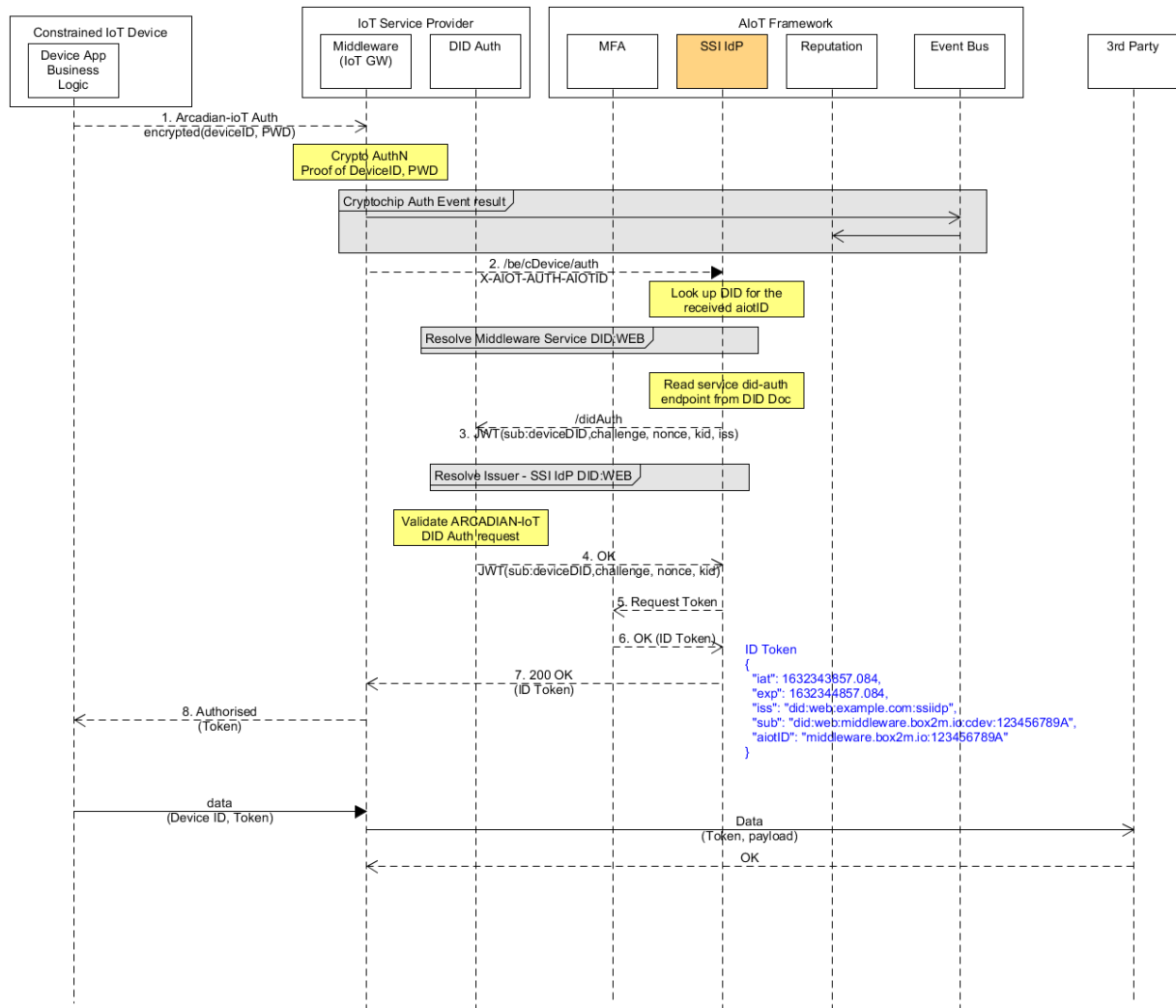


Figure 36 Authenticate a Constrained IoT Device with its Public DID

3.1.3.3.11 Service Provider registers a Person in the ARCADIAN-IoT framework (P1)

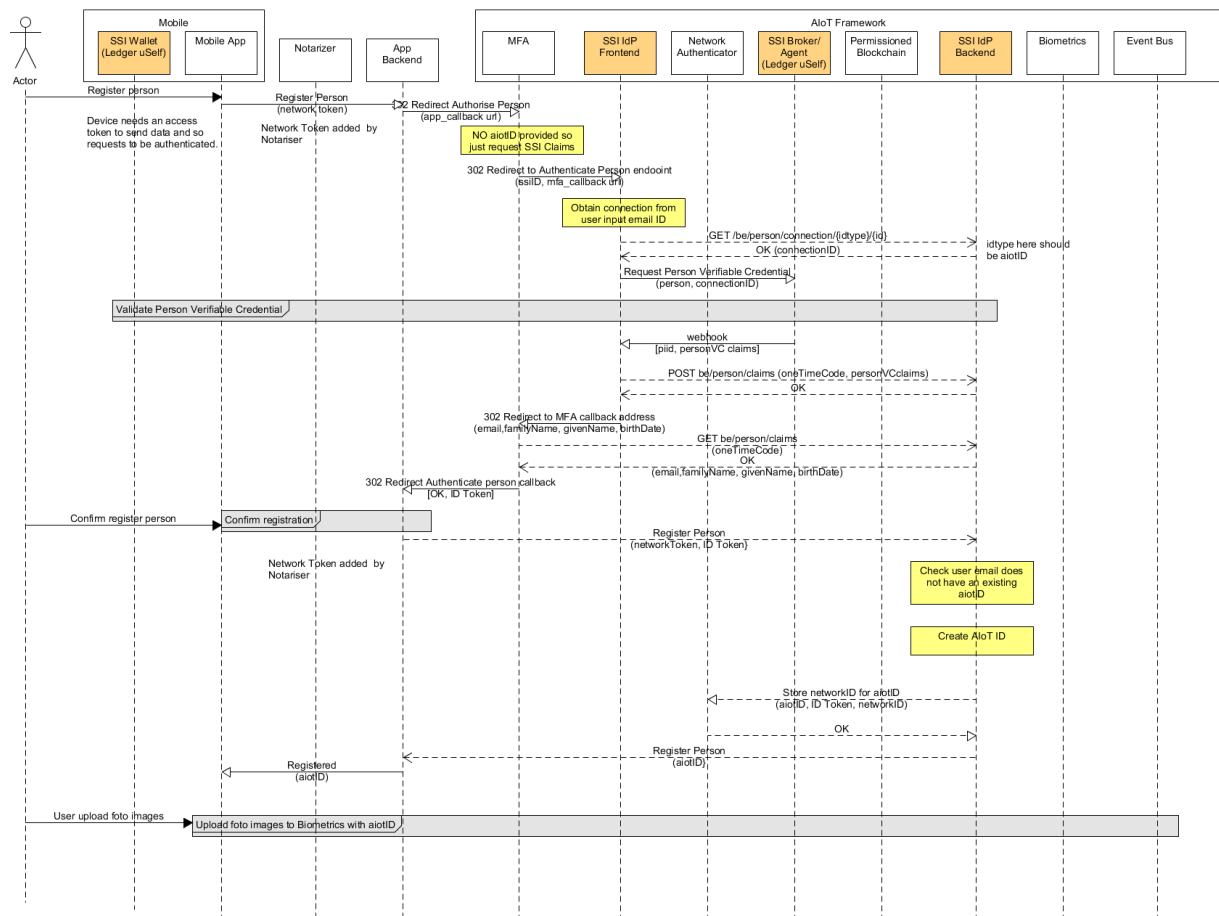


Figure 37 - Service Provider service registers a Person

3.1.3.3.12 Service Provider registers Organization Member in the ARCADIAN-IoT framework (P2)

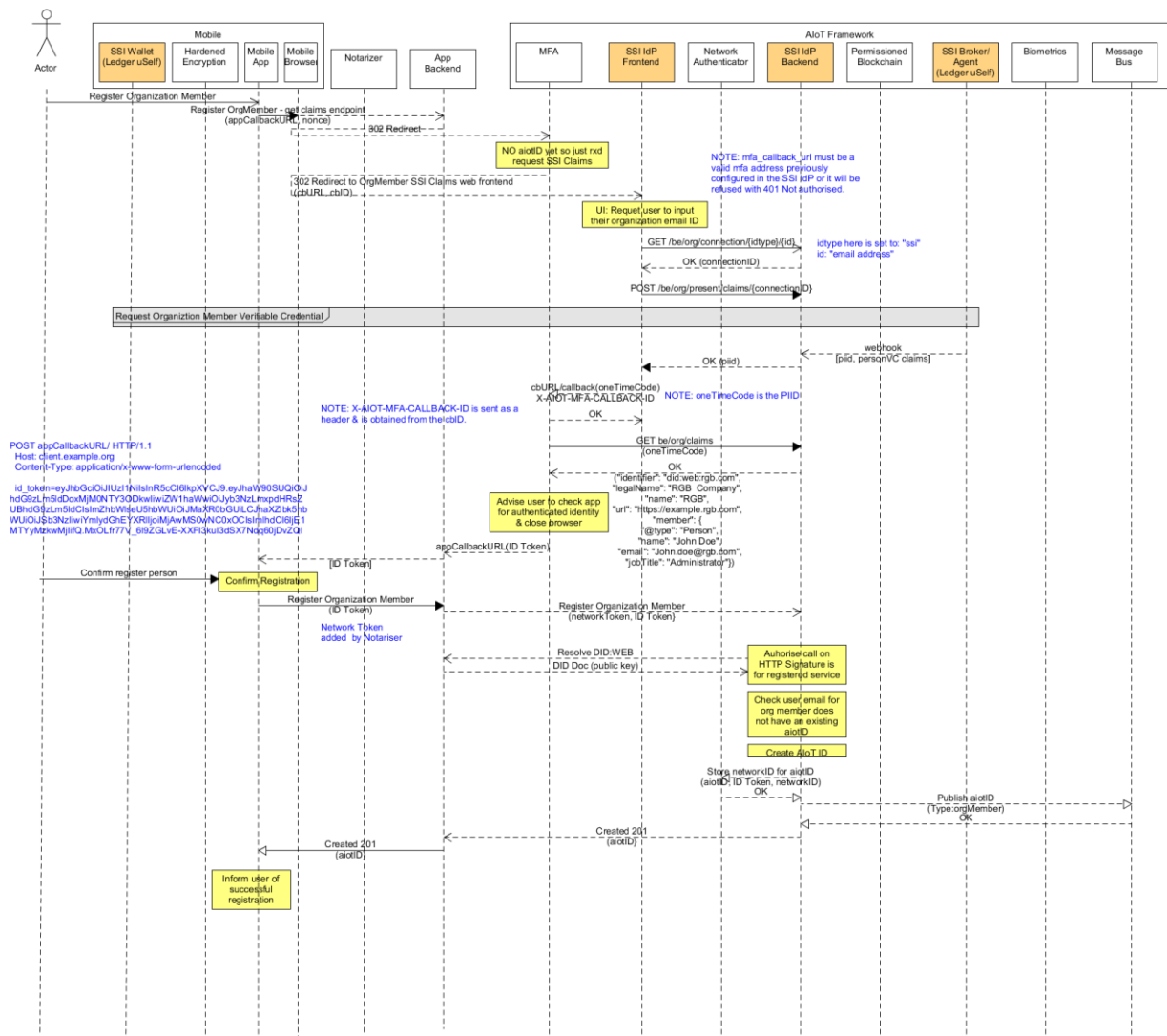


Figure 38 - Service Provider service registers an Organization Member

[illegible]

3.1.3.3.14 Service Provider updates a registered Person, Organization Member or IoT-Device identity in the ARCADIAN-IoT framework (P2)



3.1.3.3.15 Service Provider deletes registered person, Organization Member or IoT-Device from the ARCADIAN-IoT framework (P2)

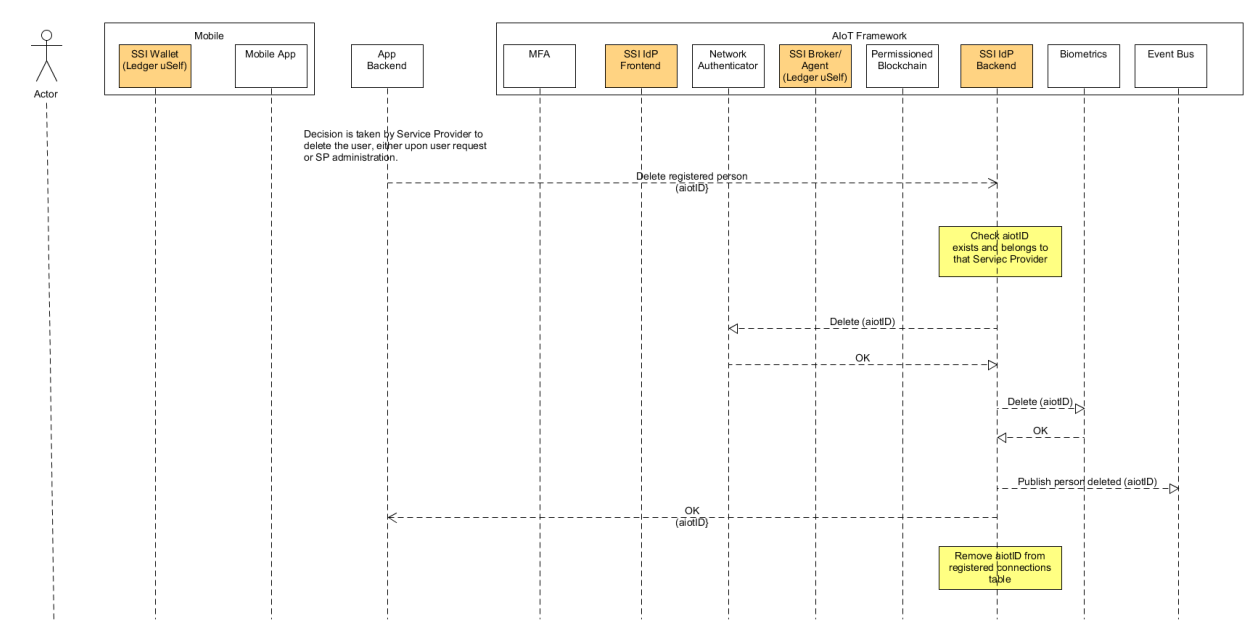


Figure 40 - Service Provider deletes a registered Person that it previously registered

3.1.3.4 Interface description

The external interfaces to ARCADIAN-IoT components and Service Provider systems are exposed from the SSI IdP. The SSI IdP interface description is shown in the following figure and published on the project's Gitlab³⁸.

ARCADIAN IOT SSI module API Specification

0.0.23 OAS3

This document contains the formal specification of the interfaces with SSI IdP component in the ARCADIAN-IoT Framework.

Servers

/

Authorize

SSI IdP Backend - Constrained IoT Devices

Constrained IoT Devices Identity Management Backend

POST

/be/cdevice/register

Register an IoT Device in AIOT Framework.

PUT

/be/cdevice/register/{aiotID}

Update to a registered constrained IoT Device in AIOT Framework.

DELETE

/be/cdevice/register/{aiotID}

Delete the ARCADIAN-IoT ID provided by the Service Provider service. It is checked that the aiotID prefix is belonging to that Service Provider service.

POST

/cdevice/authenticate

Request to authenticate a constrained IoT Device with an aiotID.

³⁸ https://gitlab.com/arcadian_iot/verifiable-credentials/-/blob/main/ssiIdP/interface/ssiIDPopenapi.yml

SSI IdP Backend - IoT Devices Devices Identity Management Backend

POST	/be/device/register	Register an IoT Device in AIOT Framework.	✓	🔒
PUT	/be/device/register/{aiotID}	Update to a registered IoT Device in AIOT Framework.	✓	🔒
DELETE	/be/device/register/{aiotID}	Delete the ARCADIAN-IoT ID provided by the Service Provider service. It is checked that the aiotID prefix is belonging to that Service Provider service.	✓	🔒

SSI IdP Backend - Organizations Organization Identity Management Backend

POST	/be/org/create	Administrator can add to the organisation member table with other members details	✓	
POST	/be/org/issue/{personalIdentifier}	BE Request to be issued with an Organisation member VC to the user's wallet.	✓	
GET	/be/org/connection/{idtype}/{id}	BE Request to get the connectionID depending on the identifier given i.e. aiotID or sslID, where sslID could be email (for Organization Member)	✓	
POST	/be/org/present/claims/{connectionID}	Request user to present Org Member claims for the users SSI wallet connection.	✓	
GET	/be/org/{legalNameBase64URL}/{id}	BE Request to get Org Member claims by querying the Org and email Id so to get issued with a Org member VC	✓	
GET	/be/org/claims/{oneTimeCode}	BE Request to get authenticated Org Member claims by querying the oneTimeCode received in the event and corresponds to PIID	✓	
POST	/be/org/register/	Register an Organization Member in AIOT Framework.	✓	🔒
PUT	/be/org/register/{aiotID}	Update to a registered Org Member in AIOT Framework.	✓	🔒
DELETE	/be/org/register/{aiotID}	Delete the ARCADIAN-IoT ID provided by the Service Provider service. It is checked that the aiotID prefix is belonging to that Service Provider service.	✓	🔒
POST	/org/authenticate	Request to authenticate an Org Member with an aiotID.	✓	

SSI IdP Backend - Persons Person Identity Management Backend

POST	/be/person/create	BE Request to be issued with a self-attested Verifiable Credential based on user's verified email address.	✓	
POST	/be/person/create/ssi/connection/{email}	BE Request to create a DID connection invitation with the user's wallet.	✓	
GET	/be/person/connection/{idtype}/{id}	BE Request to get the connectionID depending on the identifier given i.e. aiotID or sslID, where sslID could be email (for person) or DID (for devices).	✓	
POST	/be/person/present/claims/{connectionID}	Request user to present Person claims for the users SSI wallet connection.	✓	
GET	/be/person/claims/{oneTimeCode}	BE Request to get authenticated Person claims by querying the oneTimeCode received in the event and corresponds to PIID	✓	
POST	/be/person/register/	Register a person in AIOT Framework.	✓	🔒
PUT	/be/person/register/{aiotID}	Update to a registered person in AIOT Framework.	✓	🔒
DELETE	/be/person/register/{aiotID}	Delete the ARCADIAN-IoT ID provided by the Service Provider service. It is checked that the aiotID prefix is belonging to that Service Provider service.	✓	🔒
POST	/person/authenticate	Request to authenticate a person with an aiotID.	✓	

SSI IdP Backend - Service Service Identity Management Backend

POST	/be/service/register	Register an SP service in AIOT Framework.	✓	🔒
DELETE	/be/service/register/{aiotID}	Delete the ARCADIAN-IoT ID provided by the Service Provider to delete an SP Service. It is checked that the aiotID prefix is belonging to that Service Provider.	✓	🔒

Figure 41 - SSI IdP Interface description

3.1.3.5 Technical solution

3.1.3.5.1 Deployment architecture view (optional)

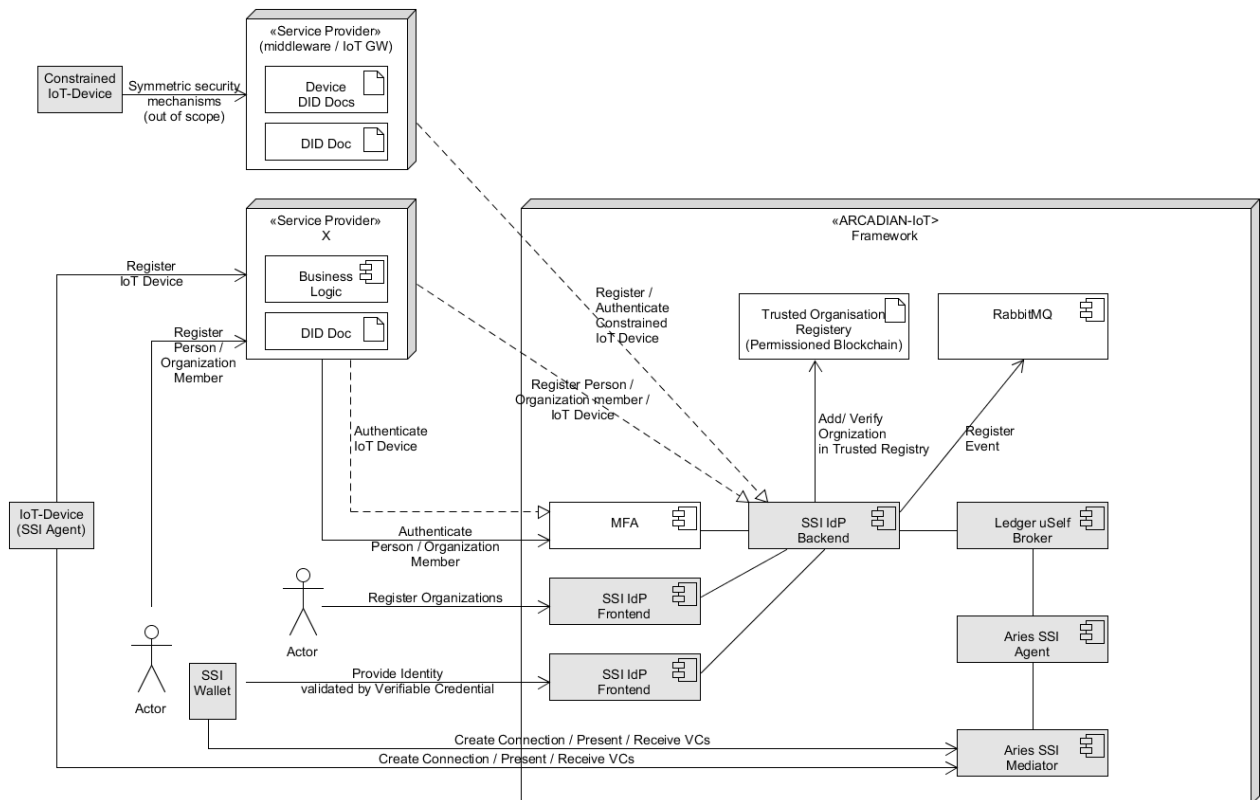


Figure 42 - Self-Sovereign Identity deployment in the ARCADIAN-IoT Framework

3.1.3.5.2 Domain model

The domain model shown below is for handling the following registered ARCADIAN-IoT entities:

- Persons,
- Organization Members
- IoT-Devices (*device*)
- Constrained IoT-Devices (*cDevice*)
- Service Provider Organisations
- Service Provider Service

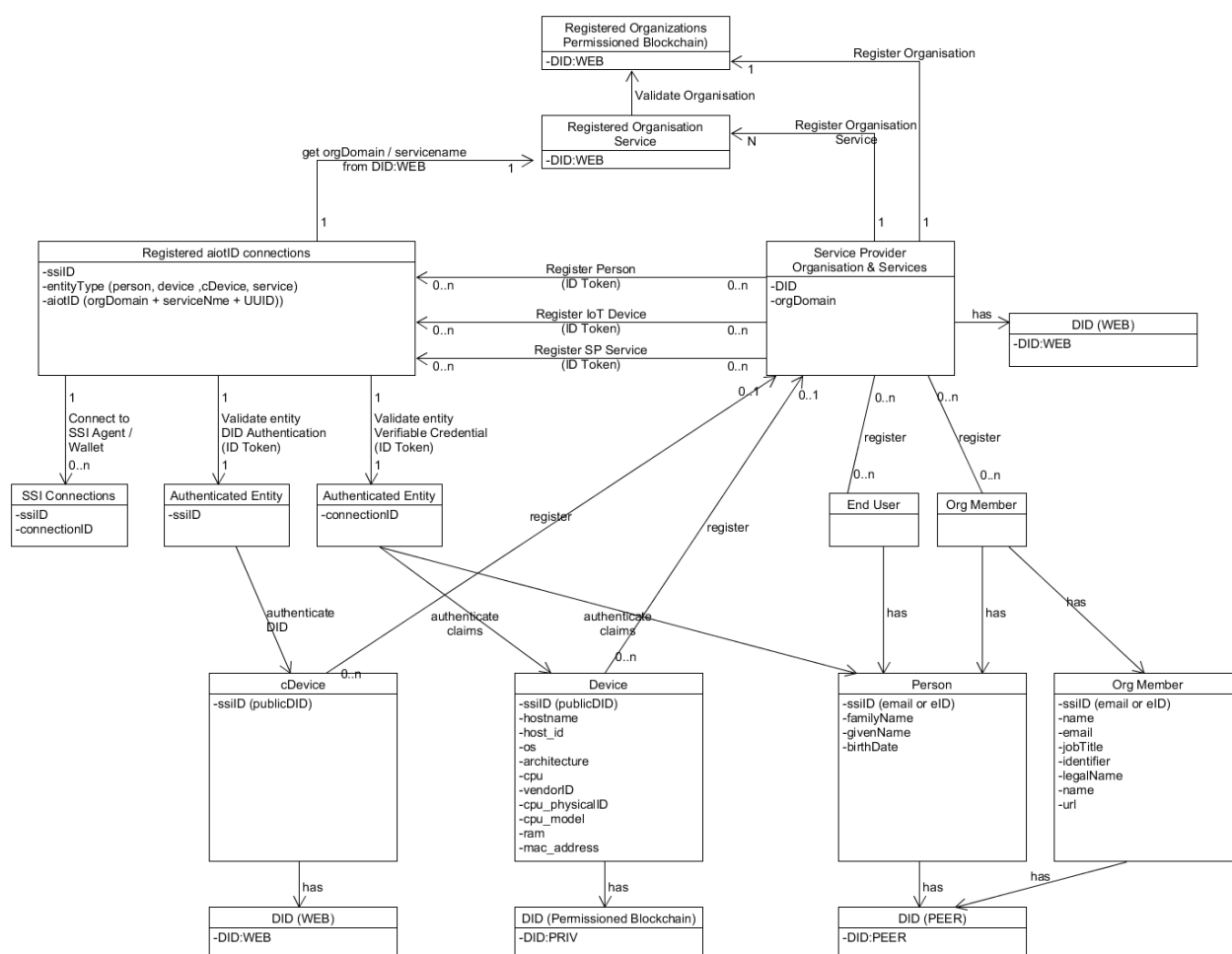


Figure 43 - SSI IdP Registered Entities

3.1.3.5.3 API specification

The Open API specification for the SSI IdP is uploaded to ARCADIAN-IoT GitLab project³⁹:

3.1.3.5.4 Frontend design

SSI IdP Issuer

The frontend supports the issuing of Person's and Organization Member Verifiable Credentials (see section 3.1.3.1.1) to end users of the service as a pre-requisite to register with any service with ARCADIAN-IoT. In a real-world scenario, the user is expected to already have a national eID issued to their wallet as per the use cases envisaged for the new European Identity Wallet [44].


The frontend supports sending the invitation to the user's email so to establish a connection from the user's mobile SSI Wallet to the framework's SSI Agent and be issued with a Person Verifiable Credential. The screenshot below shows the issuer screen obtaining the user's details.

Additionally it supports the issuing of Organization Member Verifiable Credentials (see section 3.1.3.3.2), noting that the person must first authenticate with valid Person VC to be able to request an Organization Member VC, to check if the company has that person in their database.

³⁹ https://gitlab.com/arcadian_iot/verifiable-credentials/-/blob/main/ssildP/interface/ssildPopenapi.yml

localhost:4200/personcreateform-component

Person Credential



e-mail

Name

Family Name

Birth date

Figure 44 - SSI IdP Issuer Screen

The next screen is an example of the presentation of the QR Code that is scanned by the Ledger uSelf mobile SSI Wallet to make a connection with the framework's SSI Agent.

A screenshot of a web browser window. The address bar shows a localhost URL. The page title is "Person Credential". Below the title bar, there is a blue header area. In the center of the page, there is a large QR code. Above the QR code, the text "Please open LedgerUserf app and scan the invitation" is displayed.

Figure 45 - QR code display to connect to the SSI Agent

SSI IdP Authentication

As regards person authentication, the SSI IdP Frontend receives redirects from the Multi-Factor Authentication (MFA) component (see section 2.4) to request a user to present their Verifiable Credential so to provide the MFA with authenticated identity claims from a user's SSI wallet.

The MFA can include in the request to the SSI IdP Frontend the `aiotID`, in which case the user just needs to confirm the request is for him/her and open their mobile wallet to provide their credential. Alternatively, if the `aiotID` is not provided by the MFA component in its request to the frontend, as per the registration flow or if the `connectionID` is not found, then the frontend will ask the user to identify themselves by their verifiable credential identity, which is an email address in the pilots rather than a real `eID`.

3.1.3.5.5 *Ledger uSelf mobile SSI wallet*

The following figure shows a screenshot of the mobile SSI Wallet with an example of a Person Verifiable Credential issued from the SSI IdP.

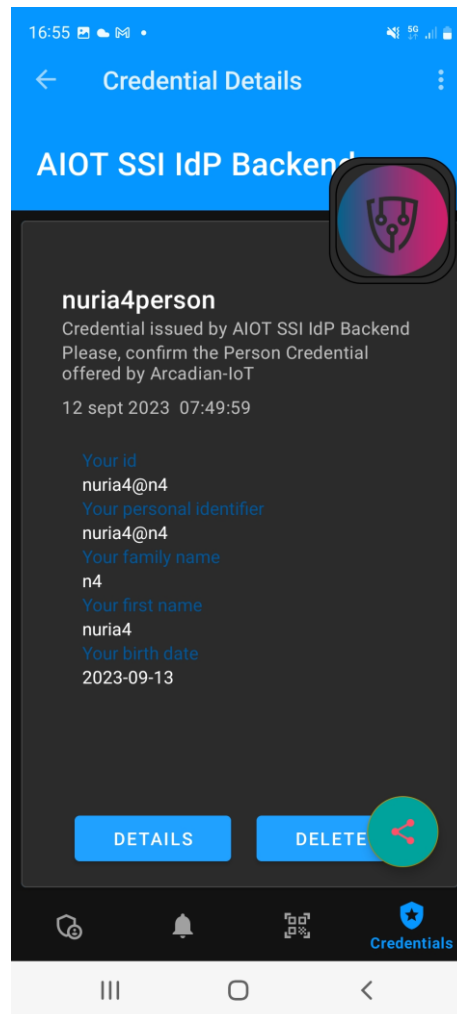


Figure 46 - Ledger uSelf mobile SSI Wallet UI

3.1.3.5.6 SSI IdP Backend design

The SSI IdP backend functions are described below.

Issuer Person Verifiable Credential

As a pre-requisite for registering users, it is needed for users to be issued with an identity Credential to their mobile SSI Wallet, and this will be issued by the ARCADIAN-IoT SSI Agent. This is aligned with eIDAS natural person schema defined for VerifiableID by EBSI [51].

A non-normative example of the Person Verifiable Credential issued by the framework's SSI Agent is given below:

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  id: "https://example.com/credentials/4jkj9",
  type: ["VerifiableCredential", "VerifiableAttestation", "VerifiableId"],
  issuer: "did:ebis:zgPs5MVWHwJJb4g9kZvYf3e",
  issuanceDate: "2021-11-01T00:00:00Z",
  validFrom: "2021-11-01T00:00:00Z",
  credentialSubject: {
    id: "did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGpbnnEGta2doK",
    personalIdentifier: "john.doe@example.com",
    familyName: "Doe",
    firstName: "John",
    dateOfBirth: "1939-10-01",
  },
  credentialSchema: {
```

```

    id:"https://api-pilot.ebsi.eu/trusted-schemas-
    registry/v2/schemas/0x0f41a1f4934e11f8f98e444ed34fb040645dcc04cead2ae28fca062eabfac181",
    type: "FullJsonSchemaValidator2021",
  },
  expirationDate: "2031-11-30T00:00:00Z",
  issued: "2021-10-30T00:00:00Z"
},
"proof": {
  "type": "Ed25519Signature2018",
  "created": "2021-11-13T18:19:39Z",
  "verificationMethod": "did:elem:ledgeruseful.eu:#key-1",
  "proofPurpose": "assertionMethod",
  "jws": "z58DAdFfa9SkqZMVPxAQpic7ndjjhghjhjSayn..1PzZs6ZjWp1CktyGesjuTSnmnm76mmwRdo
  WhAfGFCF5bppETSTojQCrfFPP2oumHKtz"
}
}

```

Person , IoT Device Authentication

The SSI IdP backend is responsible for persisting the connections the framework's SSI Agent has with the SSI Identity associated with the entity, which in the case of persons in the prototype is the user's email address and the public DID for IoT Devices.

Additionally, the SSI IdP will maintain a table of all registered entities with the ARCADIAN-IoT ID, the SSI ID and the entity type (e.g. Person, IoT-Device).

Therefore, entity authentication with the SSI Agent / Wallet will support the following scenarios:

- 1) During onboarding the authentication request from the MFA will not include the aiotID and thus the frontend will request the backend to request the Person / Device VC from the user's SSI Wallet / Agent, based on the entity's SSI ID. In this case the backend will lookup the connection persisted for that SSI ID to then request the Person / Device VC to the SSI Wallet / Agent.
- 2) When the aiotID is included in the request from the MFA, the request for the backend from the frontend will include the aiotID and the backend will lookup the SSI ID associated with that aiotID before requesting the Person / Device VC for the associated connection.

SSI Webhooks

Communications with the mobile SSI Wallet / Agent is asynchronous and as such the SSI IdP subscribes to the Broker to receive notifications. The following notifications are the main ones that concern the SSI IdP:

- upon accepting a connection
- upon accepting an issued Verifiable Credential
- upon presenting any Verifiable Credential

A non-normative example of a Webhook notification is as follows:

```

{
  "id": "string",
  "topic": "didexchange_states",
  "Message": {
    "ProtocolName": "didexchange",
    "Message": {
      "@id": "string",
      "@type": "https://didcomm.org/didexchange/1.0/complete",
      "~thread": { "thid": "string" },
      "~transport": { "~return_route": "all" }
    },
    "Properties": {
      "connectionID": "string",
      "invitationID": "string"
    }
  }
}

```

```

    },
    "Type": "post_state",
    "StateID": "completed"
  }
}

```

Person Registration

As part of person or IoT Device registration a new ARCADIAN-IoT Identity (aiotID) is requested to the framework to be created and managed per Service Provider. ARCADIAN-IoT identity is managed within the framework as follows:

- A specific Service Provider is responsible for registering their users to the ARCADIAN-IoT framework.
- The same end users can be registered by other Service Provider to ARCADIAN-IoT, each with a different aiotID, so the end user can have multiple aiotIDs: one per Service Provider.
- When an SP wants to delete its user and all its associated data it only removes the ARCADIAN-IoT component data associated with that aiotID registered by that SP service.
- If during a registration attempt a user is found to have an existing ARCADIAN-IoT ID associated with their SSI ID for the same SP the registration request will be rejected.
- During registration the SSI IdP will provision the Network Authorization component with the aiotID and received network token.
- During registration the SSI IdP will provision the Biometrics component with the aiotID and received face image.

The ARCADIAN-IoT identity is therefore specified as follows:

```
"aiotID": "orgDomain:UUIDv4"
```

The "orgDomain" is obtained from the registered DID WEBS of the Service Provider (see section 2.1.3.2.3) and the UUID v4 is generated upon registration.

A non-normative example is:

```
"aiotID": "ATOS.net:b666ca65-0faa-4e8b-a4bb-b5db253dd878"
```

IoT Device Registration

Registration of IoT Devices follows the same way that is carried out for Person Registration described above.

Service Provider Organization Registration

It is assumed that Service Provider organizations will already have well known public Decentralized Identifiers such as did:web. In ARCADIAN-IoT each participating organization must create and host did:web DIDs for their organizations to be able to use the ARCADIAN-IoT framework.

Organization's that are part of the ecosystem must publish their Public DID and organization information to the Trusted Organization Registry on the permissioned blockchain (see section 4.3 in D3.3 [47]), or request another organization to do this on their behalf.

During organization registration an organization object is stored off-chain on the Permissioned Blockchain to provide a trusted registry of all registered organisations in the ARCADIAN-IoT ecosystem, and the hashes of this are stored on-chain to provide the root of trust.

A non-normative example is given as follows:

```
{
  "did": "did:web:atos.net"
  "dataObject": {
    "name": "Atos",
    "legalName": "Atos",
    "url": "atos.net",
    "email": ""
  }
}
```

Service Registration

Once a Service Provider organisation is registered in the ARCADIAN-IoT Framework it can register the services that will make use of the ARCADIAN-IoT Framework (i.e. its component services) for its end user Persons and IoT-Devices.

When a Service Provider registers a service it will create an aiotID and publish it to the event bus, and store the registered service DID with its aiotID.

Data Model

The sslID backend will keep a record of all registered ARCADIAN-IoT Identities and SSI connections to user SSI wallets and IoT Devices SSI Agents as per the following data model.

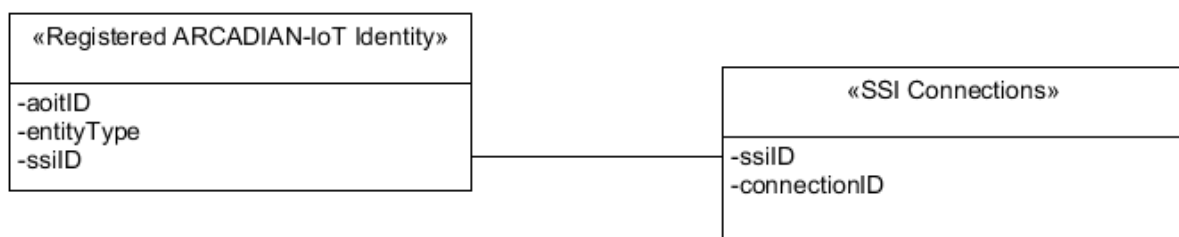


Figure 47 - SSI IdP Data Model

Additionally, please refer to the Domain Model in section 3.1.3.5.2.

RabbitMQ

Upon registration of a Person or IoT Device the SSI IdP will publish a registration event to RabbitMQ with the following identity attributes:

- aiotID
- entityType
- sslID
- ssiClaims{}

3.1.3.5.7 Ledger uSelf Broker

The ledger uSelf Broker deployed in P1 is the background prototype Broker, developed in Kotlin, as described in section 3.1.2.1.

To meet the needs of ARCADIAN-IoT deployment of the Broker and particular flows for IoT Devices is developed in GO so to be able to integrate it with the Hyperledger Aries GO Agent in one package that can be deployed as an executable directly in IoT Devices as well as a Docker

image on servers. This provides a leaner implementation for more efficient operation in IoT Devices with reduced resources, as compared with the typical cloud resources.

3.1.3.5.8 Security aspects

The security between the components has not been implemented in the project and thus relies upon the framework components deployed in the same domain behind a firewall.

This is noted as pending for future work to raise the TRL level of the ARCADIAN-IoT Framework.

Note: Mutual TLS was the initial candidate technology to provide trusted and secure communications between the ARCADIAN-IoT framework components, but there are also other means including those employed in microservices architectures.

3.1.4 Evaluation and results

The first prototype P1 provided:

- An SSI Issuer to Issue Person Verifiable Credentials
- An SSI Wallet for a person to manage their Verifiable Credentials on their mobile device
- An SSI Verifier to verify the presentation of Person VCs to the framework
- Integration with the MFA to return an ID Token to the Service Provider with the person claims presented
- An SSI IdP frontend and backend to support registration and authentication to the ARCADIAN-IoT framework for persons

The final prototype P2 provided:

- An SSI Issuer to Issue Person, Organization Member & Device Verifiable Credentials
- An SSI Wallet for a person and Organization Member to manage their Verifiable Credentials on their mobile device
- An SSI Agent developed in GO to be deployed on an IoT Device to manage its DID and Verifiable Credentials
- A single SSI Verifier to verify the presentation of Person, Organization Member & Device VCs on behalf of the ARCADIAN-IoT framework
- Integration with the MFA to return a token to the Service Provider for the verified entity i.e. Person, Organization Member, IoT Device and constrained IoT Device
- An SSI IdP frontend and backend to support registration and authentication to the ARCADIAN-IoT framework for Persons, Organization Members, IoT Devices and constrained IoT Devices

Test validation has been carried out internally by ATOS and component integration and pilot validation is carried out in WP5.

3.1.5 Future work

Future work, beyond the ARCADIAN-IoT project, for the Verifiable Credentials component is listed as follows:

- To add support for SIOP & OpenID4VP protocols so to align with OpenID protocols that has been announced during this project for Data Spaces convergence and the European Digital Identity Wallet.
- To implement selective disclosure in a new version of the wallet that makes use of the BBS+ signature to provide ZKP privacy mechanisms. The signature is supported as

provided by Task 4.1, however it is needed to implement supporting UI on the wallet to let the user decide what claims to provide.

- Investigate use of eSIM for public / private key to support Decentralized identifiers in IoT Devices.
- Provide a security mechanism to establish trust between the framework components other than deployed in the same domain behind a firewall

3.2 Authorization: Network-based Authorization enforcement and authorization distribution

3.2.1 Overview

3.2.1.1 Description

ARCADIAN-IoT has different technologies that relate with authorization processes. Specifically, there is the Network-based Authorization enforcement, which includes authorization information distribution to devices, based on entities (e.g. devices) trustworthiness information or security reputation. The framework includes as well a Self-aware Data Privacy component, where users (or IoT Service Providers) define business-related or role-based authorization rules to allow access to their data.

In this section we will focus on the component that both enforces trust-based authorization rules in the cellular network core (between devices and internet services) and also informs devices' secure element (the SIM, in the case) of the trustworthiness level of the device where it is located.

3.2.1.2 Requirements

The requirements for this component are the following⁴⁰:

- **To provide a dynamic Network-based Authorization enforcement based on entities trustworthiness level and security policies:** A network-based enforcement tool, positioned at the core of the cellular network provider, to control entities (persons or devices) communication/interaction based on their security reputation or trustworthiness.
- **To distribute authorization information to devices' secure element:** Ability to securely distribute information about devices' trustworthiness to their hardware secure element, for enabling actions of self-protection and self-recovery.

3.2.1.3 Objectives and KPIs

The Network-based Authorization enforcement and authorization distribution component contributes to the accomplishment of the following objectives and KPIs.

KPI scope	
This component contributes to the objective of <i>providing distributed and autonomous models for trust, security and privacy – enablers of a Chain of Trust (CoT)</i> . The contribution is based on the enforcement of the defined model for trust, security, and privacy, being this component a relevant autonomous agent able of receiving inputs from ARCADIAN-IoT Reputation System to enforce security actions. In this sense, the main objective of this component is to research and develop a novel process for communication authorization enforcement, according to entities trustworthiness level.	
Measurable Indicator	
1. Automatic bidirectional communication authorization enforcement for devices and people according to trustworthiness levels and its dynamic changes related with security events (Y/N) 2. Time to enforce the authorization policy after the network being informed	
Target value	Achieved value
1. Dynamic bidirectional communication enforcement according to security policies and trustworthiness level 2. Near real time	1. Target achieved 2. Target achieved (<1 second)

⁴⁰ Requirements enhanced since the last public deliverables according to the research done.

KPI scope	
This component also contributes to the objective of <i>self and coordinated healing with reduced human intervention</i> , by informing the SIM of device's trustworthiness information, triggering the subsequent SIM-based protection and recovery actions. In this case the research focuses on informing the SIM of devices trustworthiness level for it to be able to act as an enabler of self-protection and self-recovery ⁴¹ .	
Measurable Indicator	
1. Ability to securely inform the eSIM of devices trustworthiness level (Y/N) 2. Use of eSIM in device self-protection and self-recovery actions (Y/N) 3. Number of different devices where the innovation is demonstrated	
Target value	Achieved value⁴²
1. Y	1. Target achieved
2. Y	2. Target achieved
3. At least 2	3. Target achieved

3.2.2 Technology research

To accomplish the research objectives the work had the following phases:

- (1) Define a unified vision over the trust information being communicated with the partner responsible for ARCADIAN-IoT Reputation System, which will be directly integrated with this component, as the Reputation System will provide to the Network-based Authorization the trustworthiness information and security policies necessary to its autonomous action.
- (2) Research on how to create a core network testbed able of accommodating the Network-based Authorization component. This network testbed should allow, in a first stage, to test the technology with virtual devices. In a second stage and for the demonstration of the technology, the core network testbed should connect real devices with real networks, mediated by the Network-based Authorization.
- (3) Have a working prototype of the component with security rules being enforced automatically in the core network according to each device trustworthiness.
- (4) Have a mechanism to securely distribute the authorization information to devices secure element, which, accordingly, will trigger processes of self-protection or self-recovery.

Depicting the functional understanding of the Network-based Authorization component, it leverages the trustworthiness (i.e., the security reputation) of the entities communicating into the network to enforce communication authorization rules. The simplest high-level example of this process is the one of a device that for some reason the Reputation System finds that it may be leaking private data. In this case, the Reputation System may request the Network-Based Authorization to not allow outgoing communication from the device (keeping incoming communication open for, e.g., Self-Recovery actions, like firmware updates).

A more concrete example of the full functional flow that shows the integration of the Network-based Authorization in an ARCADIAN-IoT scenario is the following:

At a given moment, and for no expected reason, drones A, B and C, all from the same brand and model, which were just turned on to be available to provide Drone Guardian Angel (DGA) service, start sending an unusual and very high amount of data (e.g. high resolution live and continuous video of its surroundings) to DGA backend services. While performing their service, these drones are also sending the data, decrypted, to an unknown internet service. If more drones have the same behaviour, this can cause service

⁴¹ The technology applies to all SIM forms (e.g. SIM, eSIM and iSIM).

⁴² Further details in the evaluation section.

degradation or even a full outage, potentially being a Denial-of-Service (DoS) attack to DGA service. Also, sending data decrypted to an unknown service may be seen as a serious security or privacy breach targeting DGA business or its users' private data. These behaviours hamper the IoT devices' (drones in this case) ability to perform their service, due to the quite suspicious behaviour and to the very high battery consumption.

ARCADIAN-IoT Behaviour Monitoring and Flow Monitoring detect these suspicious behaviours and inform the Reputation System of these events. Considering the threats posed by the drones and the existent security policies, the Reputation System reduces their trustworthiness to the lowest rating possible.

*At that moment, automatically, the **Network-based Authorization** is informed of the devices security reputation changes and applies to their communication policies the rules related to the lowest rating possible, e.g., that they cannot communicate with external services or receive communication from any service. When a recovery strategy is defined, if needed to reduce human intervention, the Reputation System can request the Network-based Authorization to allow the communications for the recovery of the device. The same component also triggers information to the devices' secure element (to ARCADIAN-IoT applet in the SIM profile), for them to take protective measures. From this moment on, drones A, B and C cannot continue overloading DGA services nor send decrypted data to the unknown internet services.*

*After components like device Self-Protection mitigate the threats, and after all self-recovery processes are successfully taken at the devices, the Reputation System is informed. According to the defined trustworthiness rules, the devices' reputation is set to a trustworthy level again. At the same time, automatically, the **Network-based Authorization** mechanism redefines the communication policies for these devices, to allow them to have normal communication again, informing as well the devices' secure element that the device is trustworthy again.*

This scenario allows to understand the expected actions and interactions of the Network-based Authorization component within ARCADIAN-IoT framework, when applied in the specific domain of surveillance in smart cities with drones. However, the component, as all the framework, aims to be agnostic to the IoT solution and the same principles apply to the contexts of, e.g., smart grid monitoring or medical IoT.

3.2.2.1 Background

3.2.2.1.1 Adding *trust*-related policies to a core network and Open5gs

In today's network architectures, communication authorization, the related policies, and billing are already a focus point. 3GPP's Policy and Charging Control (PCC) architecture⁴³ provides access, resource, and quality of service (QoS) control⁴⁴ to mobile networks. Two components of this architecture are the Policy and Charging Rules Function (PCRF) and the Policy and Charging Enforcement Function (PCEF) for up to 4G, or the Policy Control Functions (PCF), an evolution of the PCRF/PCEF for 5G.

PCRF acts as the policy manager of the network, the central point of decision that provides policy control and flow-based charging control decisions. The PCEF usually lives in the serving gateway,

⁴³

<https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=810>

⁴⁴ <https://www.netmanias.com/en/post/techdocs/10997/lte-pcrf/policy-and-charging-rules-function-pcrf-in-lte-epc-core-network-technology>

can offer packet inspection capabilities, and enforces the rules provided by the PCRF. Besides these two components, an Application Function (AF) interacts with other applications and services that require a dynamic PCC⁴⁵. 3GPP's PCC architecture describes an AF as “an element offering applications that require dynamic policy and/or charging control over the IP CAN (IP Connectivity Access Network) user plane behaviour”. The AF extracts session information and media-related information from the application signalling and provides application session-related information to the PCRF using the Rx⁴⁶ protocol. This information is the part of the inputs used by the PCRF for the Policy and Charging Control Decisions and the rules engine can be triggered by one of these messages⁴⁶.

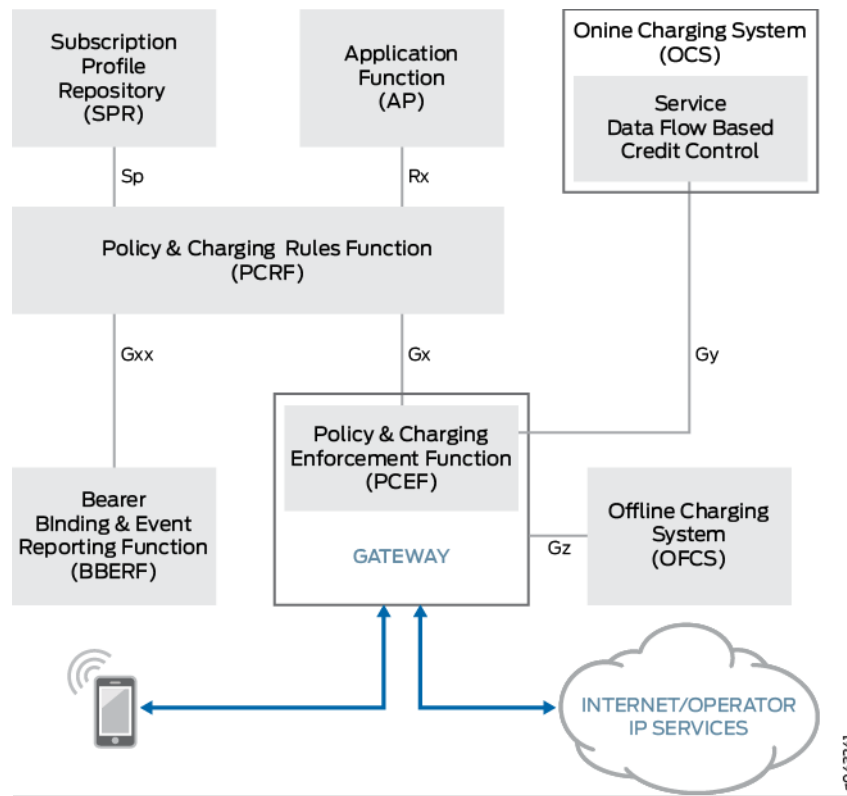


Figure 48 - 3GPP's PCC Architecture overview⁴⁵

Although the Rx interface is DIAMETER-based, efforts have been made by the 3GPP to provide a RESTful approach, with XML as the content body format, to these functions. In this case, a Protocol Converter (PC) acts as the middleman between the AF and the RX-speaking PCRF⁴⁷.

It is worth noting that there are two types of PCC rules, predefined and dynamic. The former is already set up in the PCEF and can only be activated or deactivated by the PCRF, while the latter can be provisioned by the PCRF via Gx interface to the PCEF⁴⁸ and can be activated, modified, and deactivated in runtime.

⁴⁵ <https://www.juniper.net/documentation/us/en/software/junos/subscriber-mgmt-sessions/topics/topic-map/3gpp-policy-charging-control-provisioning-accounting.html>

⁴⁶ <https://www.netmanias.com/en/post/techdocs/10997/lte-pcrf/policy-and-charging-rules-function-pcrf-in-lte-epc-core-network-technology>

⁴⁷ <https://www.3gpp.org/more/1629-rx-interface>

⁴⁸ <https://www.netmanias.com/en/?m=view&id=techdocs&no=11863>

In the context of ARCADIAN-IoT, PCRF/PCEF solutions can be leveraged to an efficient and dynamic route and prioritization of the network traffic⁴⁴ as a means of providing trust-based authorization inside the network. PCRF/PCEF use policy-based authorization, however, as seen in ⁴⁹, it is possible to build a mixed authorization system that joins static and dynamic policy-based authorization with different rules' sources. This system combines the several factors to create a flexible authorization framework. To implement the Network-based Authorization, an analysis of network implementations was carried out. The main aspects considered were the presence of an API that allowed to manipulate the subscriber information and the related communication policies, but also, if possible, a free and open-source solution.

Nowadays there are some solutions that fit this purpose, including Open5GS, Magma, srsEPC, to name a few. The current choice, as testing hypothesis for the network testbed is Open5GS. Open5GS⁵⁰ "is a C-language open-source implementation of the 5th Generation Core (5GC) and Evolved Packet Core (EPC), i.e. the core network of New Radio/Long-Term Evolution (NR/LTE) network". It supports the current 3GPP's PCC architecture described before (although without the OCS and OFCS, which are not relevant to our needs) and also the new 5GC service-based architecture where the policy is handled by the Policy Control Function (PCF). Open5GS can be installed in Ubuntu through the package manager, but it also supports other Linux-based operating systems by building it from the source code. It can be also run in a dockerized environment or even in AWS, making it a great candidate for core network testbed choice.

Figure 49 shows Open5GS architecture, where its capabilities as core network testbed are depicted. In what regards the policy control functions (control plane), indispensable for implementing the Network-based Authorization, it is possible to identify both a PCF and a PCRF, which indicates that Open5GS can operate both in 5G and in previous generations of broadband cellular network technology (e.g. 4G).

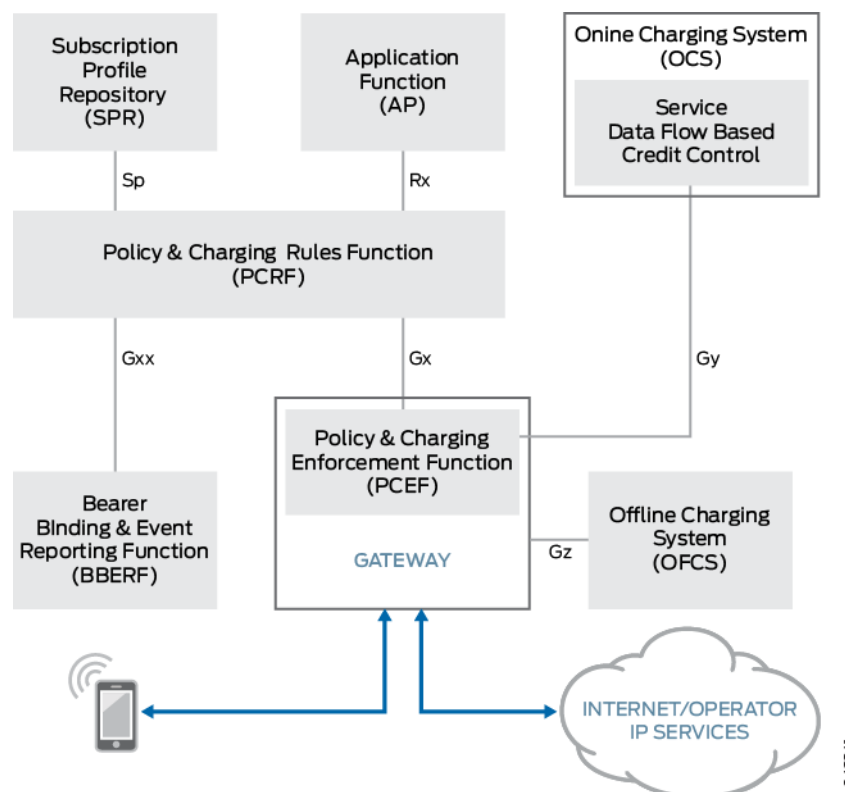


Figure 49 - Open5GS architecture⁵¹

⁴⁹ <http://rewerse.net/publications/download/REWERSE-RP-2005-116.pdf>

⁵⁰ <https://open5gs.org/>

⁵¹ <https://open5gs.org/open5gs/docs/guide/01-quickstart/>

Open5GS is supported by an active community and provides various resources that have been source of knowledge for the current work:

- Website with blog, tutorials, and other documentation (<https://open5gs.org/>).
- GitHub repository⁵² with source code, issue tracker and discussion board.

Discord server with a community chat room (invitation link is provided via GitHub's readme).

3.2.2.2 Research findings and achievements

The main research findings and achievements are the following:

- A vision and architecture for the use of core network functions for trustworthiness-based communication authorization enforcement.
- The deployment of network core testbed (4G and 5G) based on an open-source implementation (Open5GS) to test, evaluate and demonstrate the authorization component.
- Integration of production networks with the testbed, for integrating partners' real cellular devices communicating to Internet services, and test of the authorization component with them. For demonstration purposes and traffic routing to the testbed, a private network was built and configured for use in ARCADIAN-IoT project.
- The Authorization Enforcement component that, according to trust information received from the Reputation System, automatically allows or denies the communication for specific devices (outgoing and incoming communication controlled independently). The authorization enforcement is applied in near real time, after the receipt of the trust information from the Reputation System.
- A secure process for distributing trust information for the devices' secure element, involving the SIM security applet and a new GSM OTA service, integrated with the Authorization Enforcement component.

In terms of enforcement of authorization policies, it was found that, in the network side, the bitrate manipulation could play an important role. By manipulating this common networking parameter for each subscriber, the intended reputation-based security actions could be taken. The following examples exemplify how.

- *Uplink bitrate set to 0 to a subscriber found to be leaking private data:* by setting the uplink bitrate to 0, the communication from that subscriber to the internet is not authorized. Therefore, this action would stop potential attacks exploring that privacy vulnerability.
- *Downlink bitrate set to 0 to a subscriber found to be accessed by unauthorized entities:* by setting the downlink bitrate to 0, the communication from the internet to that subscriber is not authorized. Therefore, this action would stop potential attacks exploring that vulnerability that allowed unauthorized access.
- *Uplink bitrate set to 5% of the normal bitrate to a subscriber suspected of being part of a DDoS:* In some cases, there are no certainties of the level of compromise of a device, and in some cases just blocking the connectivity can damage a business or even harm a life (e.g. in medical IoT scenarios). In these cases, reducing the bitrate will slow down a potential attack, without completely cutting off the connectivity.
- *Upon recovery of a secure state, uplink or downlink automatically set back to 100% of the normal bitrate:* if or when security measures are taken in a device that mitigate the vulnerabilities that led it to a state of unauthorized communication (e.g. downlink bitrate of 0), its normal connectivity state can be recovered by setting the bitrate to 100% again.

Several combinations of the above may be used according to the security strategies intended. For example, in some cases it may be acceptable to, after having cut the connectivity of a

⁵² <https://github.com/open5gs/open5gs>

subscriber, activate it again when the protection measure is defined (e.g. a firmware update over the air). This would happen with the Reputation System informing the Authorization component that the device should be able to communicate again, according to the recovery strategy. Such an action can be relevant for reducing human intervention in the recovery of compromised devices.

Apart from the bitrate manipulation, the authorization information distribution to devices' secure element also plays an important role in stopping some types of attacks. For example, if a device is found to be poisoning one or more IoT data lakes or performing impersonation attacks, the Authorization component can request the SIM from that device to stop the digital signatures to the payloads being sent, allowing the system to act accordingly (e.g. not consider messages not signed by the SIM). In this case the SIM (with ARCADIAN-IoT security applet) is used as a security agent and not just as a connectivity enabler – the mentioned protection action applies even if the device tries to use other communication technology other than cellular networks (e.g. WiFi).

In all cases, in ARCADIAN-IoT, the reputation-based policy will be provided to the Network-based Authorization component by the Reputation System, ensuring no human intervention in this process.

The whole setup was successfully tested in several stages and for several cases. There were internal tests with virtual and real devices; integration tests with the partners technologies (Reputation System) and within concrete use cases with the 3 IoT solutions present in the project.

3.2.2.3 Produced resources

The resources produced in the context of this component are:

1. The architecture including the integration interfaces with the Reputation System.
2. A cellular core network testbed integrated with production networks, with a private network configured (network side and device side) for the devices used within the project.
3. The Network-based Authorization component, deployed in the testbed able of, for each ARCADIAN-IoT-ID, enforce the communication policies for the related network subscriber, according to trust policies provided by the Reputation System.
4. A GSM OTA service for communication with the SIMs according to the standards. This channel is integrated with the Network-based Authorization component, which will trigger the communication to the SIMs according to trust policies provided by the Reputation System.
5. A SIM software (within the security applet built in the context of ARCADIAN-IoT Hardened Encryption component) to make it ready for receiving trust information and act accordingly.

3.2.3 Design specification

3.2.3.1 Logical architecture view

Figure 50 depicts this component' logical architecture. In ARCADIAN-IoT, we leverage existent cellular network policy enforcement tools to enable novel mechanisms of dynamic communication authorization. The innovation is that authorization is expected to be enforced according to the entities' trustworthiness level and security policies provided by ARCADIAN-IoT's Reputation System. This Reputation System will be informed by other ARCADIAN-IoT components of security-related parameters that influence entities trustworthiness (e.g. Behaviour Monitoring or Remote Attestation), passing this information to the components that may act according to trustworthiness changes. In this sense, the Network-based Authorization component will be informed of each entity trust information and security policy to apply, and automatically translate it in knowledge understandable by the cellular networks' functions that can act accordingly. By

using PCFs or PCRFs/PCEFs, mechanisms known for their scalability and performance, programmatically orchestrated with the security-based Reputation System, this component automatically acts in the presence of threats or vulnerabilities, e.g. by blocking sensitive data leakage to the internet or unauthorized control of devices behaviour from an internet service.

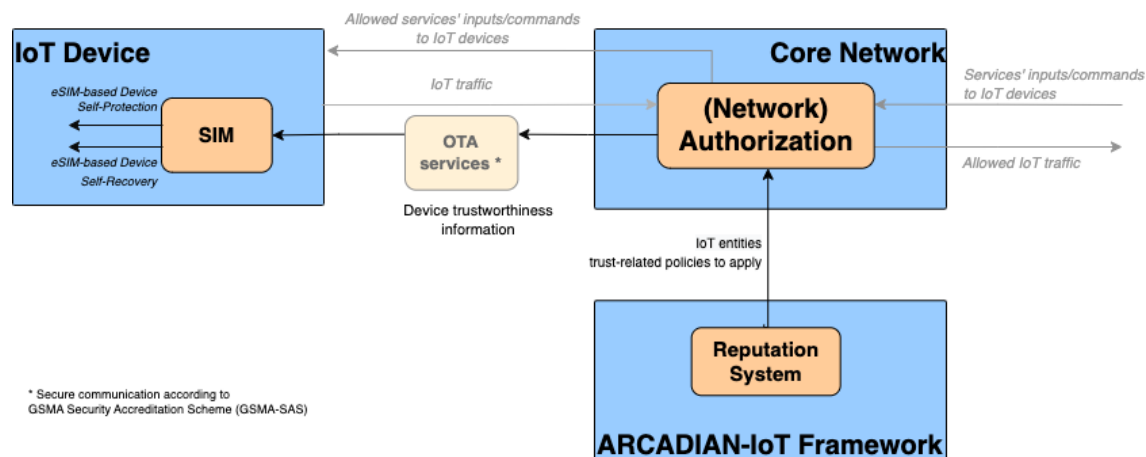


Figure 50 - ARCADIAN-IoT Network-based Authorization high-level architecture

Lastly, with the knowledge of entities (e.g. devices) trustworthiness information, the component positioned securely in the core of 4G or 5G networks, therefore between devices and internet services, will securely inform devices secure element (eSIM) regarding the devices level of compromise. This will allow the ARCADIAN-IoT's security applet to take actions of protection or recovery according to devices' trust level. More details about these specific actions can be found in ARCADIAN-IoT's Hardened Encryption component reporting.

3.2.3.2 Interface description

As described in previous section, the Network-based Authorization component will interact directly with the Reputation System and with the SIM security applet. The interfaces are the following (the Network-based Authorization component should be seen as a consumer of the content exchanged):

Table 7 - Network-based Authorization interface to external components

Component	Communication type	Content exchanged
Reputation system (publisher)	RabbitMQ AMQP 0.9.1 (authenticated publish/subscribe queue)	Trust related policies to apply to a given ARCADIAN-IoT ID
eSIM	GSM OTA services	Device / application trustworthiness level

3.2.3.3 Technical solution

To depict the current technical solution, the architecture shown in Figure 51, extends the previously analysed Figure 50, focusing now the technical details of the Network-based Authorization component.

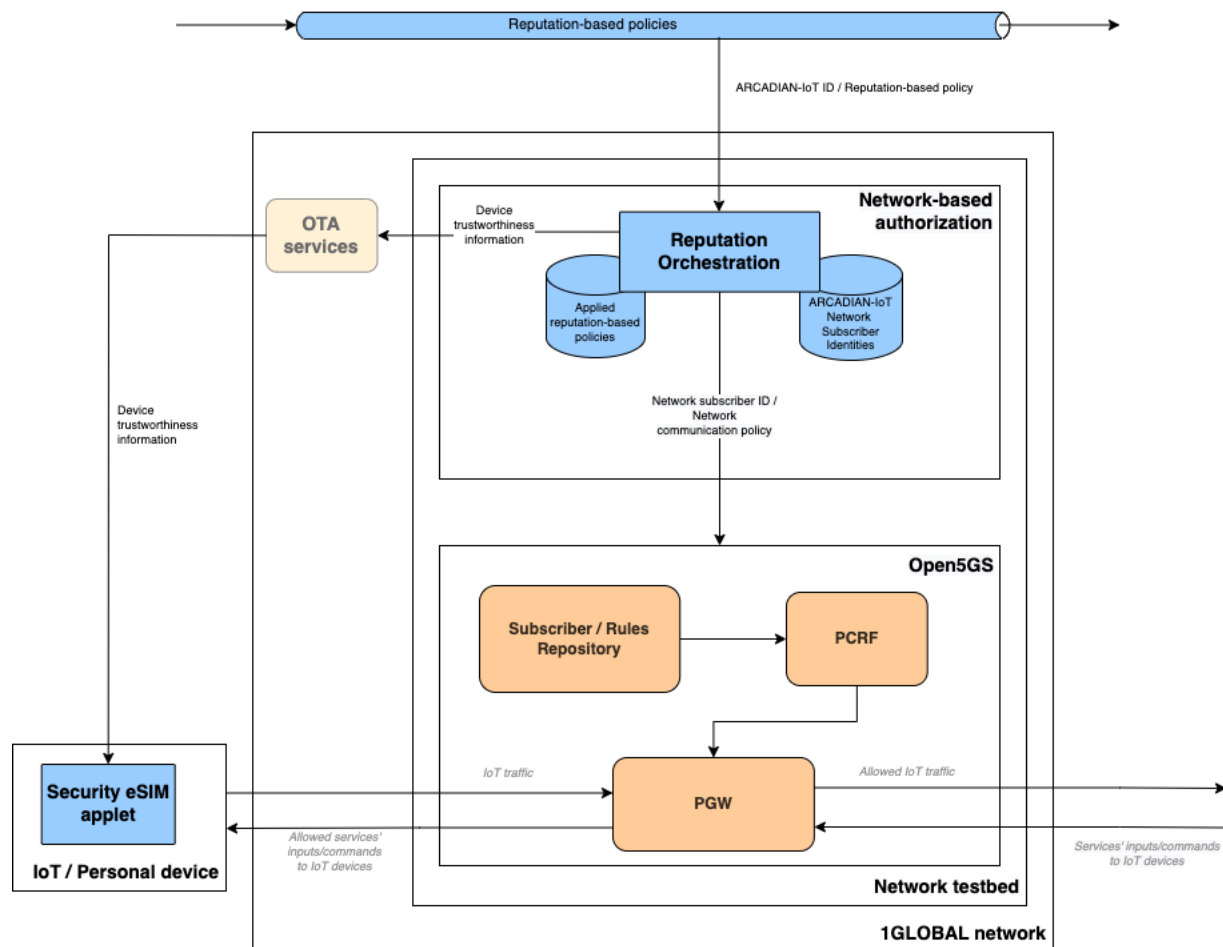


Figure 51 - Network-based Authorization technical architecture

The integration vision defined by the directly involved partners (1GLOBAL for the Network-based Authorization, and UC for the Reputation System) assumed that the communication between the Reputation System and Network-based Authorization should be made in a publish-subscribe approach, with the Reputation System publishing reputation trust-related policies to apply to a particular ARCADIAN-IoT entity (i.e., which communication rules should be applied to devices/people/services according to their reputation score).

As can be seen in Figure 51, both the Open5GS (open-source implementation of 4G / 5G core network functions) and ARCADIAN-IoT Network-based Authorization component form the intended network testbed, which has been deployed in an AWS environment.

The Network-based Authorization component has a major subcomponent named *Reputation Orchestration*, whose functions are:

- To receive and store reputation-based policies to apply to a given ARCADIAN-IoT ID
- To consult the network subscriber identity (e.g. IMSI) that matches the received ARCADIAN-IoT ID
- To generate network communication policies, understandable by Open5GS, according to the reputation policies received; and forward these policies and the related network subscriber ID to be enforced in Open5GS.
- To distribute authorization information to devices' security SIM applet (for self-protection and self-recovery actions), according to the reputation policies received.

3.2.3.3.1 API specification

No external APIs exist in the Network-based Authorisation beyond the interfaces specified in Table 7.

In what regards the exchange between the Reputation System and the Network-based Authorization the structure of the information sent by the former is the following:

- Arcadian-IoT ID
- Action: Reputation-based action to apply (e.g. block communication)
- Information to SIM: If it should be requested an action of Self-recovery or Self-protection to the SIM

Even though the component is ready to enforce communication authorization uplink and downlink separately, for the moment and according to the research done by the Reputation System, the control of the communication will always happen in both directions.

Concerning the information sent via GSM OTA to the SIM security applet, as mentioned, it depends on the received reputation information, which may request the RoT action of Self-protection or of Self-recovery. As there are only these 2 options and no other information is needed, the APDU (Application Protocol Data Unit) sent to the ARCADIAN-IoT SIM applet will have a form of a flag that informs if the action of Self-Protection or Self-Recovery should be taken.

3.2.4 Evaluation and results

The Network-based Authorization component contributes to ARCADIAN-IoT objective of *providing distributed and autonomous models for trust, security and privacy – enablers of a Chain of Trust (CoT)*. Its contribution is based on the enforcement of the defined model for trust, security, and privacy, being this component a relevant autonomous agent able of receiving inputs from ARCADIAN-IoT Reputation System to enforce security actions.

In this sense, the main objective of this component was to research and develop a novel network-based process for communication authorization enforcement, according to entities trustworthiness level. We assess this objective as being successfully achieved. An automatic and bidirectional communication authorization enforcement for devices and people according to trustworthiness levels and its dynamic changes related with security events was developed, integrated with the Reputation System, and successfully tested within the IoT solutions of the consortium partners. The time to enforce the authorization policy after the network being informed was near real time.

This component also contributes to ARCADIAN-IoT objective of *self and coordinated healing with reduced human intervention*, by informing the SIM of device's trustworthiness information, triggering SIM-based protection and recovery actions. In this case the research focused on a process for informing the SIM of devices trustworthiness level for it to be able to act as an enabler of self-protection and self-recovery⁵³. This objective was also achieved. The ability to securely inform the SIM of devices trustworthiness level and use it in device self-protection and self-recovery actions was developed and tested. This innovation was integrated and successfully tested in lab and within 2 IoT solutions of the consortium partners (vigilance with drones in smart cities and medical IoT), with all the devices available. It is being demonstrated in the 3 IoT solutions from the consortium in WP5.

3.2.5 Future work

The immediate future work for ARCADIAN-IoT Network-based Authorization, is its demonstration and dissemination of its application in the IoT solutions targeted in WP5. The final definition regarding its exploitation path is also in progress in the next few months, being the defined exploitation actions put in place after the project.

⁵³ The technology applies to all SIM forms (e.g. SIM, eSIM and iSIM).

3.3 Reputation System

3.3.1 Overview

3.3.1.1 Description

The Reputation System component in ARCADIAN-IoT determines the reputation values – score associated with the entities in the ARCADIAN-IoT framework – persons, devices and services. The reputation score represents the trust information regarding a certain entity, and such information is built based on data received from other entities and services in the domains use cases. Different reputation models are considered to build the score: a) the alpha-beta model; b) alpha-beta with severity and c) the dominance relationships.

3.3.1.2 Requirements

The requirements of the reputation system have been documented in D2.4:

- Requirement 5.3.1 – **Information of Entities identification**: The entities interacting with the system need to be known by the reputation system. Such entities include persons, IoT devices, and application/services.
- Requirement 5.3.2 – **Information of Entities interactions**: The interactions of the diverse entities are input for the reputation score. Such interactions can be intra- device or inter-entities.
- Requirement 5.3.3 – **Trustable storage mechanisms for reputation**: The reputation system requires mechanisms to store the reputation of entities in a distributed and trusted fashion, without single point of failure.
- Requirement 5.3.4 – **Service registration in the reputation system**: Services should register in the reputation system and/or provide information of entities interactions in pre-configured topics of the reputation system (e.g., Device and Network IDS events received from device Behaviour Monitoring and Network Flow Monitoring components, respectively).

3.3.1.3 Objectives and KPIs

The work of the reputation system is decomposed into the key objectives:

- Determine reputation score of entities interacting with the ARCADIAN-IoT framework in the diverse domains.
- Support storage of reputation scores in a distributed and reliable fashion.
- Support the sharing of reputation information with components interested with the reputation information.

As documented in D2.4 the main KPIs associated with the reputation system are threefold: (1) Number of messages analysed per unit of time: Messages indicating interactions between entities; (2) Time required to determine reputation; (3) Types of entities supported by the reputation system, at least 3 types. These are detailed in the following tables.

KPI scope
Determine Reputation Score
Measurable Indicator
Number of messages analysed per unit of time
Benchmarking (OPTIONAL)

Events received through the message bus and processed per unit of time	
Target value (M30)	Current value (M30)
300 per minute or higher	300 per minute
KPI scope	
Determine Reputation Score	
Measurable Indicator	
Time required to determine reputation	
Benchmarking (OPTIONAL)	
Elapsed time since the message was received in the message bus till the determination of its score.	
Target value (M30)	Current value (M30)
In the order of milliseconds	Between 438 and 731 milliseconds
KPI scope	
Determine Reputation Score	
Measurable Indicator	
Number of entities supported in the reputation system	
Benchmarking (OPTIONAL)	
Not applicable	
Target value (M30)	Current value (M30)
3 entities	3 entities: persons, devices and services

3.3.2 Technology research

3.3.2.1 Background

This subsection documents the research in terms of reputation models, and available libraries, technologies for a scalable stream processing.

3.3.2.1.1 Reputation Models

The determination of the reputation score can rely on different models and approaches. Web services like eBay, Amazon have their own reputation models running, which normally rely on multiple mechanisms to aggregate the feedback provided by clients and users⁵⁴.

Of particular interest in ARCADIAN-IoT is the consideration of the beta distribution^{55,56}, that can consider two types of events:

- ALPHA (a) – Number of events expressed as normal (positive) behaviour. Example user performs registration in a device and provides all the required information.

⁵⁴ A. I. A. Ahmed, S. H. Ab Hamid, A. Gani, S. Khan, and M. K. Khan, "Trust and reputation for Internet of Things: Fundamentals, taxonomy, and open research challenges," *J. Netw. Comput. Appl.*, vol. 145, no. September 2018, 2019

⁵⁵ A. Josang and R. Ismail, "The beta reputation system," in Proceedings of the 15th, bled electronic commerce conference, vol. 5, pp. 2502–2511, 2002

⁵⁶ Carlos Junior et al, "A Privacy Preserving System to Consult Public Institutions Records", master thesis, University of Coimbra, 2020, <http://hdl.handle.net/10316/94061>

- BETA (b) – Number of events expressed as anomalous (negative) behaviour. As an example, the user fails to perform login after 3 consecutive times.

Besides considering the nature of events, that is if they correspond to normal or anomalous behaviour, it also includes a weight parameter that can correspond to the number of events of a specific type. The reputation score is determined considering the following equation, which determines the reputation value as a probabilistic value.

$$E(p) = \frac{a}{a + b}$$

To specify preference over the most recent interaction behaviours there is the possibility of using the Forgetting factor, where the value 0 means to consider only the most recent, while the value 1 considers all the interactions seen so far.

In the beta distribution, the feedbacks can be provided in a pair of (r,s) with a normalization weight, or as a single feedback (v), being r and s determined as illustrated in the following equations.

$$r = v \cdot w \quad s = w(1 - v)$$

The values of r and s are employed to determine a ALPHA and b BETA, respectively. The value of feedback (v) can correspond to the rating of a service in a scale of 1 to 5.

Some components in the ARCADIAN-IoT provide the severity information that is associated with an event, for instance the network flow monitor provides the severity level. Based on this data, we have modified the ALPHA BETA model to include this information, as presented below.

$$\beta_i = (\beta_{i-1} * A) + S$$

The S variable represents the severity range, which is considered according to the incident scale of Atlassian⁵⁷:

1. Value three – Highly severe
2. Value two – With medium severity
3. Value one – With low severity

More particularly, this model modifies how the BETA values, associated with negative events, are updated.

The Dominance relationship-based reputation computation (DRBR)⁵⁸ model is also of particular interest for ARCADIAN-IoT as it allows to aggregate the reputation of the diverse services, considering the information gathered from other entities, regarding a particular entity. In short, it aggregates the feedback provided by users, to a service X, considering the dominant values of reputation. The DRBR model works in several steps:

1. Identify dominance relationships, services with higher preference, or with more positive feedback
2. Model the services as a Directed Acyclic Graph (DAG), to allow choosing the services that will be ranked, in case the dominance information is not objective.

⁵⁷ Atlassian Understanding incident severity levels. <https://www.atlassian.com/incident-management/kpis/severity-levels>, 2019. Accessed on August 2023.

⁵⁸ X. Fu, K. Yue, L. Liu, Y. Feng, and L. Liu, "Reputation Measurement for Online Services Based on Dominance Relationships," *IEEE Trans. Serv. Comput.*, vol. 14, no. 4, pp. 1054–1067, Jul. 2021.

3. Considering the DAG determines the rank of services, the following equation applies for rating services:

$$r_i = \frac{\max(C) - \min(C)}{|S| - 1} \cdot (|S| - \text{idx}(s_i, RS)) + \min(C)$$

Where r_i corresponds to rating being determined for service i , RS is the ranking of services, $\text{idx}(s_i)$ is the index of service i in the RS , and C corresponds to the rating scales.

3.3.2.1.2 Stream Processing

The determination of the reputation score requires information of events, which can be received from the diverse components that are included in the ARCADIAN framework. In this regard, there is the need to be able to process data (information of events) in a scalable fashion.

Data Stream Processing Engines (DSPEs) like Apache Spark, Apache Flink, Apache Storm, Apache Samza⁵⁹ provide the foundations to process events in a scalable fashion. According to the state the art, the choice of Spark offers a set of functionalities that can be useful for the Reputation System, such as: ability to process a high number of events (higher throughput); the possibility of using in-memory storage to parse or process data according to certain filters (select only values that are higher than a threshold); or the ability to perform processing in batch or in real time fashion.

Spark can also be easily integrated in applications developed in different programming languages like Java, Python or Scala.

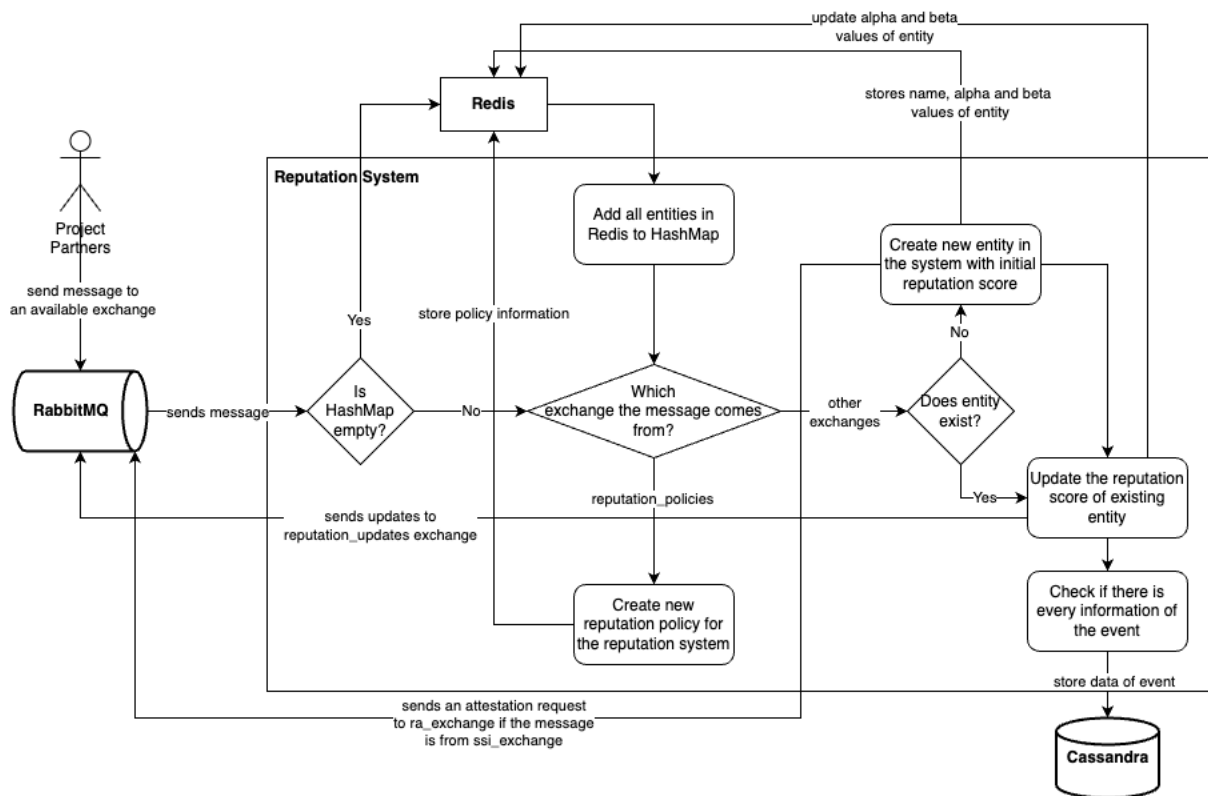


Figure 52 - Flow of processing events in the reputation system

⁵⁹ Isah, H., Abughofa, T., Mahfuz, S., Ajerla, D., Zulkernine, F., & Khan, S. (2019). A survey of distributed data stream processing frameworks. *IEEE Access*, 7, 154300–154316. <https://doi.org/10.1109/ACCESS.2019.2946884>

Figure 52 illustrates the flow of processing the diverse events that are received through RabbitMQ. Reputation System also uses Redis for reliability purposes, for instance if an instance of the reputation system fails, Redis allows to recover information of entities that were being processed before failure. Internally, the stream processing of events relies on different criteria, like the exchange that was used to present event's information, or the related policies that can be applicable. Before storing the information in the Cassandra, there is a final step that assures that all the information fields are present to store all the information associated with an event.

3.3.2.1.3 Policies and reputation

The reputation score by itself represents a value which may require additional information for the enforcement of policies. There are different approaches for the policy management. For instance, ETSI in the technical specifications TS 33.501, TS 33.117 or TS 118.103 introduces the required elements to manage policies in 5G networks. Including components responsible to keep the information of policies, others to determine the best policy to apply (PDP – Policy Decision Point), and others to really apply the policy (PEP – Policy Enforcement Point). In the ARCADIAN-IoT framework the policies are specified by the service provider.

Two types of policies are considered:

- Authorization Policies – to be applied when the authorization of device is being handled.
- Attestation Policies – to be applied to trigger the remote attestation of a device. More details are available in section 3.3.2.1.5.

In respect to the association of the reputation with authorization policies, a simple approach is followed, inspired by the iptables⁶⁰ functioning mode. The following actions can exist in policies:

- ACCEPT – accept the flow associated with a specific service, device or entity.
- DENY – do not accept the flow associated with an entity.
- THROTTLE – reduce the quality of service parameter associated with a connection, like reducing the available bandwidth.

These policies are created with the following fields:

- *aiotIDs*: An array with IDs of the domain targeted by the policy that must include the AIoT identifiers, for instance for two devices, it would be ["deviceId1", "deviceId2"];
- *infoSIM*: information to SIM that have the possible values of None, Self-Protection or Self-Recovery, these values are used by the authorization component to enforce policies;
- *action*: integer value that action type value that can be either deny (1), throttle (2) and accept (3);
- *actionRatio*: integer with the ratio to reduce bandwidth. For instance, to reduce 10\% of bandwidth;
- *minReputationScore*: float value representing the minimum reputation score that an entity must have to be affected by that policy;
- *maxReputationScore*: float value representing the maximum reputation score an entity must have to be affected by the policy.

The reputation system has the role to map the reputation scores into policies, providing this information in the reputation updates, which can be used by the authorization component. Other components also receive the reputation updates (Device behaviour monitor) but do not process the information regarding policies:

- *currentScore* –most recent reputation score
- *previousScore* – previous score that was shared

⁶⁰ <https://www.netfilter.org/>

- *entityID* – AiotID of the entity whose reputation has been updated
- *infoSIM* – with the same meaning as for the policy
- *action* – with the same meaning as for the policy
- *actionRatio* – with the same meaning as for the policy

3.3.2.1.4 Privacy aspects in reputation (storage, processing)

The analysis of the privacy aspects focused the General Data Protection Regulation (GDPR) that has been promoted by the European Union (EU). GDPR is composed by 11 chapters, but being more relevant the chapters 2, 3 and 4 for the privacy aspects of the reputation system.

Considering chapter 2 which defines several principles that must be considered regarding data management:

1. Lawfulness, fairness and transparency (on the treatment of the collected data)
2. Purpose limitation (specific, explicit, and legitimate purposes are stipulated by the controller for the processing of the data)
3. Data minimization (only the necessary data is collected)
4. Accuracy (the data should be updated and rectified or erased on subject request)
5. Storage limitation (the data is only stored while required or needed)
6. Integrity and confidentiality (security measures are applied to guarantee the security of the personal data)
7. Accountability (the controller is responsible to ensure and demonstrate compliance)

Chapter 3, on its side, focuses on the rights of data subjects, like the right to be informed, of rectification, of erasure, to restrict processing, among others. Chapter 4 defines controllers and processors, which impact the reputation system, when considering the interaction with blockchain.

Despite not being finalized, several data privacy concerns have already been identified in the perspective of using Blockchain in the reputation system, as summarized in the table below.

Table 8 - Data privacy concerns (preliminar analysis)

GDPR Principle	Data Privacy Concern
Lawfulness, Fairness	No issue as long as it is supported the informed consent by the data subject.
Transparency	May have issues if there are several channels of communication between the data subjects.
Purpose Limitation	May have issues associated, requiring a clear and transparent statement of the purpose, and access control mechanisms in place.
Data Minimization	There is a concern since the default mode of the blockchain is to append data, and the multiple copies of the ledger.
Accuracy	Blockchain assures immutability, but it is required the support for rectification and erasure of the provided data.
Storage Limitation	As blockchain append mode, some concerns regarding storage limitation apply, leading to questions like “When does the data become obsolete?”
Integrity	No concerns.
Confidentiality	Concerns can exist, depending on the mode of using the blockchain (permissioned or permissionless).

Accountability	The way blockchain is implemented can lead to issues.
----------------	---

Figure 53 summarizes the results of the conducted analysis and highlights the principles and rights of GDPR and the required technologies for a full compliance with GDPR

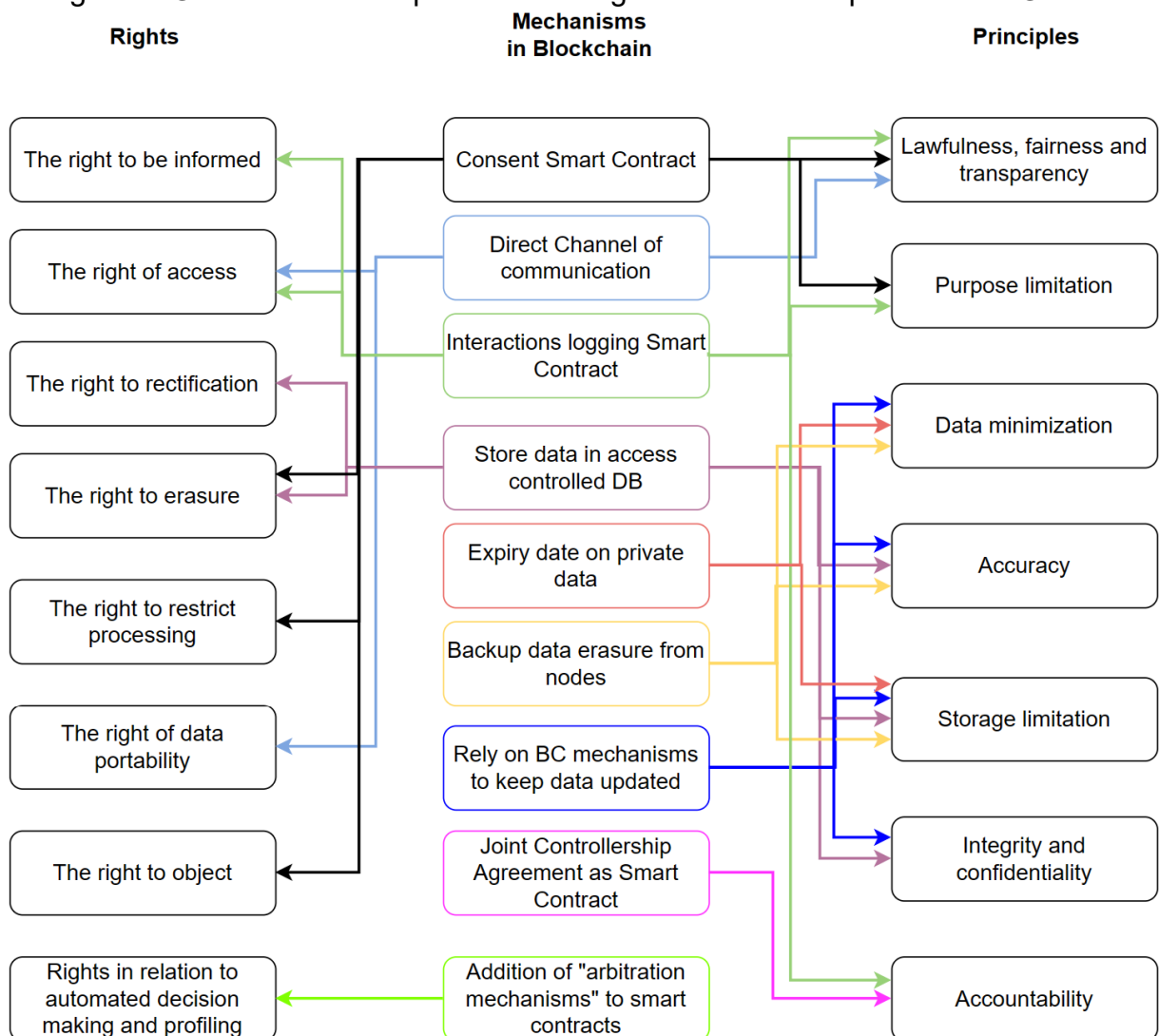


Figure 53 - Mechanisms in Blockchain for GDPR compliance

To ensure the data processing, the reputation system is connected with other components that allow the analysis of personal data, like the Personal Data Analyser (PDA), –which is able to detect the presence of Personable Identifiable Information (PII) and triggers the action of storing the data in online or offline mode. Figure highlights the components that were developed to allow this processing, as well as to enable the mechanisms in Blockchain that are required to enable the compliance of GDPR.

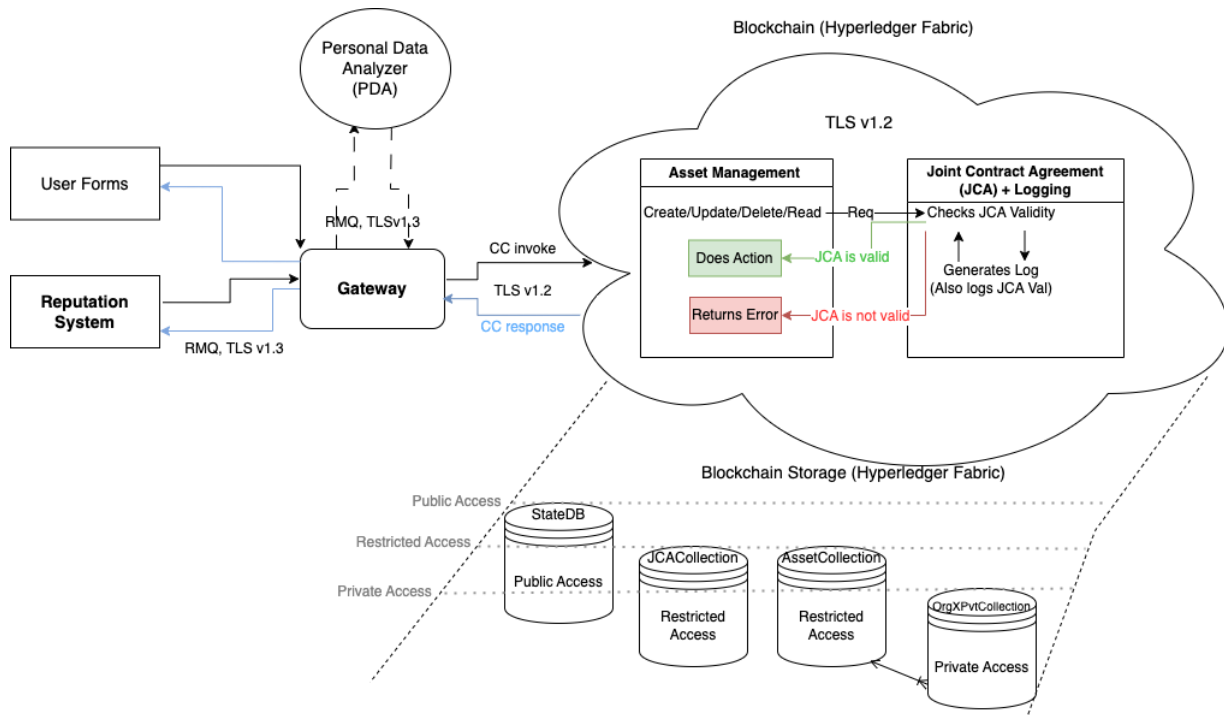


Figure 54 - Developed architecture to allow the compliance with GDPR (PDA – Personal Data Analyzer, JCA – Joint Contract Agreement)

The Gateway component analyses all the information and with the assistance of the PDA, decides the storage mode on the Permissioned Blockchain. If no private information (i.e., PII) is found in the data to be stored then the information is processed according to the Joint Contract Agreement (JCA).

3.3.2.1.5 Reputation System and its Relying Party functionality in the attestation process

Considering the Reputation System, decides - through its scoring system - whether or not entities are considered trustworthy, it has been designed to acts as a Relying Party in remote attestation procedures. Reputation System acting as a Relying Party, sends a message to the Attestation component to verify the trustworthiness of a device. Additionally the reputation system also triggers attestation as per the configured policies (e.g., when reputation decreases a certain threshold).

The attestation policies are defined with the following fields:

- *aiotIDs*: An array with IDs of the domain targeted by the policy that must include the AIoT identifiers;
- *minReputationScore*: float value representing the minimum reputation score that an entity must have to be affected by that policy;
- *maxReputationScore*: float value representing the maximum reputation score an entity must have to be affected by the policy.

Additionally, whenever the reputation system detects a new entity matching the "device" type, it requests the remote attestation component to remotely attest the device through the *ra_exchange*. As as stated before, according to the defined policies.

3.3.2.1.6 Reputation System and Visualization

A component to enable the visualization of behaviour of the reputation system was designed and implemented – The reputation system analyser. This component allows the visualization of all identities with the respective reputation score and entity type, as exemplified in Figure 55

Entity Name	Entity Type	Reputation Score ▼
wrqyi0rxb	Device	0.66666645
android	Device	0.66579634
l42nebnokl	Device	0.66315789
TV25v7JXEU	Not Yet Classified	0.63636364
q9f3aplqzq	Device	0.60000000
test@prueba.com	Not Yet Classified	0.60000000
dGVzdEBwcnVlYmEuY29t	Not Yet Classified	0.60000000
ZdbssGJVfi	Not Yet Classified	0.60000000
YXRvcy5uZXQ6MDQ1ZmRkOGQyY2I5Ny00Y2EyLTlmNTgtZmFhM...	Not Yet Classified	0.60000000
test@prueba	Not Yet Classified	0.60000000

Rows per page: 10 ▼ 1-10 of 19 |< < > >|

Figure 55 - Screen of reputation system

The combination of events, with the time dimension and type of event/action is combined in a graphic with parallel coordinates that allow the brushing, namely to highlight the events that contribute to a specific reputation value, demonstrated in Figure 56.

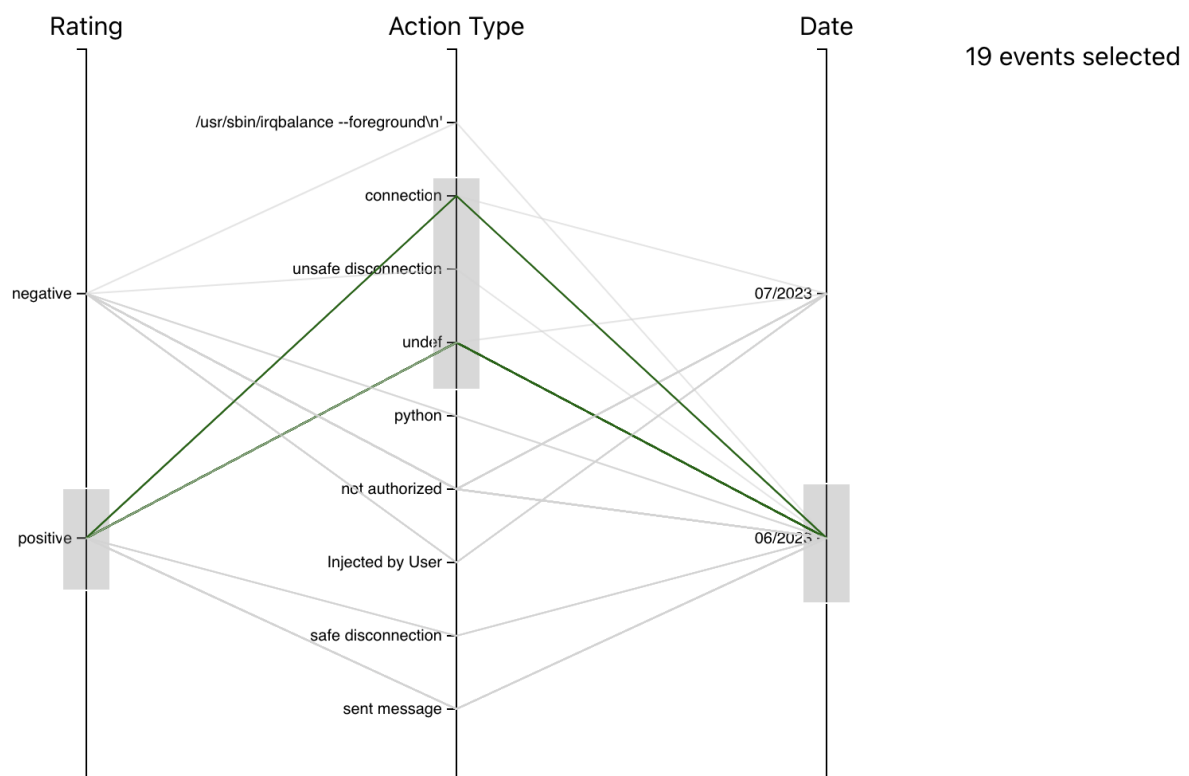


Figure 56 - Parallel coordinates chart with brushed events

3.3.2.2 Research findings and achievements

The main research findings of the reputation system include:

- The compliance with GDPR is only possible with teams with different expertise, in terms of legal aspect and technical to implement smart contracts to enforce the policies regarding data processing. The usage of external components devoted to identifying personal identifiable information is relevant.
- The processing of multiple sources of data, by itself constitutes a factor of innovation in the reputation system to determine the reputation score of an entity.
- The employment of distinct reputation models helps to measure the reputation of an entity in a more reliable fashion.
- Visualization aspects are key to help end-users to understand how the reputation system behaves, helping to increase its transparency, namely with graphics that rely on parallel coordinates and allow the comparison of reputation scores and events.

3.3.2.3 Produced resources

The produced resources include:

- The reputation system was implemented as a docker container using Java programming language, with spark libraries.
- The policy manager to allow all the CRUD of policies to associate with the reputation values. This component also relies in Java technology and provides APIs and a frontend for the interface with users (e.g., service providers or any other entity responsible for setting the policies in the target application scenario).
- Reputation System Analyser, which is a component to allow the visualization of the data and outputs associated with the reputation system.

3.3.3 Design specification

3.3.3.1 Logical architecture view

The reputation system aggregates information from several components to formulate the reputation score, as depicted in the Figure 57.

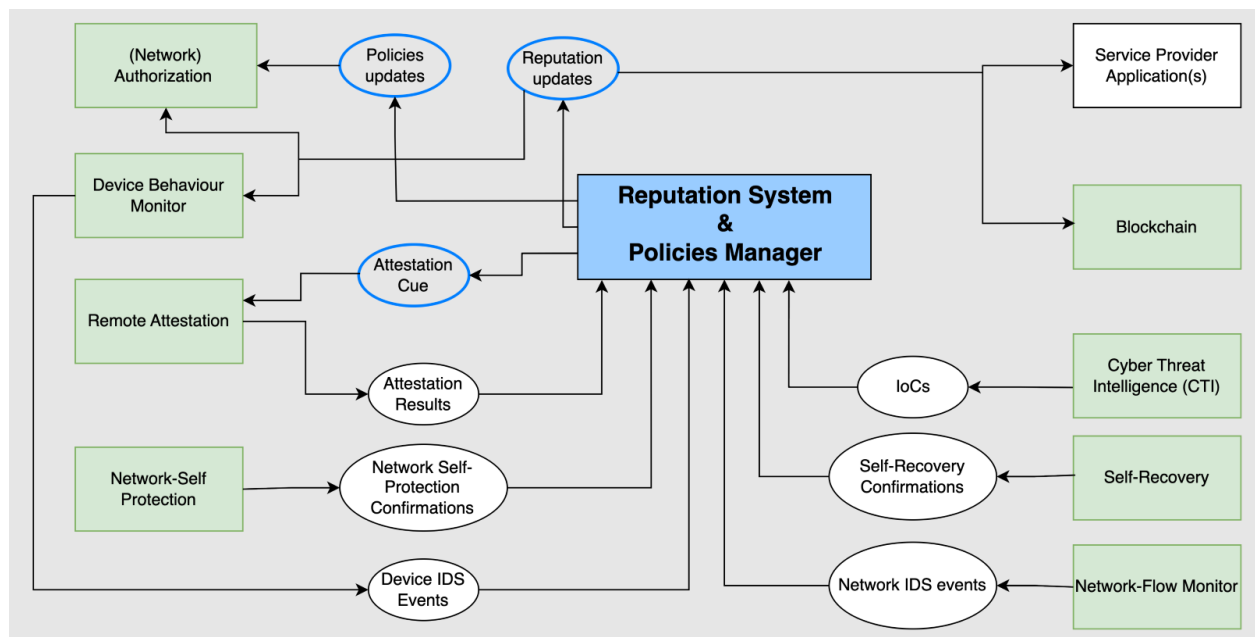


Figure 57 - Reputation System & Policy Manager logical architecture view

The reputation system receives information, from several ARCADIAN-IoT components and provides the following types of information:

- Attestation Cue
- Reputation Updates with applicable authorization policies.
- Policies Updates

3.3.3.2 Sub-use cases (Recommended)

The sub-use cases are documented in detail per domain. The analysis in each domain identifies the involved entities, how reputation can change for each one.

The following procedure is considered for the three domains (A, B, C):

- Initially the reputation is NULL for every entity
- According to the type of events that may occur, the reputation score is incremented (+) or decremented (-). This implies a classification of events according to the information associated with them.
- As per the type of events it is also determined which entity should have the reputation updated.

A exemplified analysis is provided in the appendices:

- The appendix A, documents the results for domain A
- The appendix B, documents the results for domain B
- The appendix C, documents the results for domain C

3.3.3.3 Sequence diagrams

The following diagram depicts the internal functioning of the Reputation System to determine reputation score.

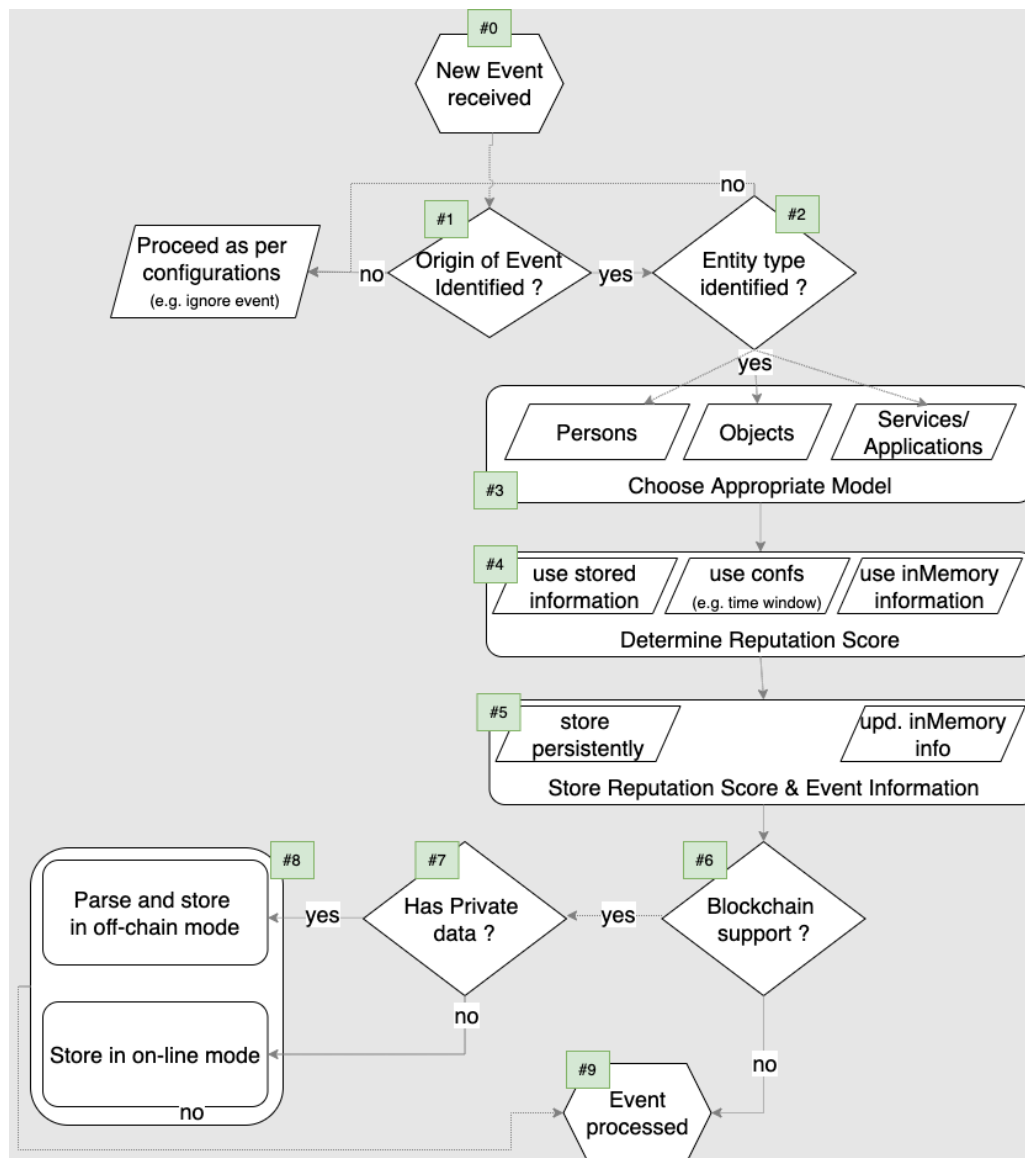


Figure 58 - Reputation System internal logic to determine reputation score

3.3.3.4 Interface description

All requests to the reputation system are handled through the RabbitMQ component. The information the Reputation System exchanges with other components is through the following queues/topics:

- Attestation Cue – To request remote attestation of a given IoT device,
- Reputation Updates – To disseminate the changes in the reputation of an entity
- Policies Updates – To disseminate the configured policies in the use case for the domain.

The policies can include authorization policies and policies for attestation.

The reputation updates are also stored in the permissioned blockchain within the respective OpenAPI interface.

3.3.3.5 Technical solution

The Reputation System is implemented in Java, the following tables document the libraries used.

Table 9 - Technologies in the reputation system and policy manager

Component	Description and third-party libraries
Reputation System	<ul style="list-style-type: none"> • Spark • Apache Commons (for alpha-beta distribution) • Cassandra • RabbitMQ • Redis
Policy Manager	Backend <ul style="list-style-type: none"> • PostgreSQL • Spring Framework • REST APIs • RabbitMQ Frontend <ul style="list-style-type: none"> • Node.JS • React

3.3.3.5.1 Reputation score range

The reputation is formulated into a score in [0,1] range. Upon the need of the score levels, there can be established as follows:

- **LOW** -> [0.0 , 0.3]
- **MEDIAN** -> [0.3 , 0.6]
- **HIGH** -> [0.6 , 1.0]

The policy manager has also a user manual to allow its employment by domain owners, as documented in Appendix D.

3.3.3.5.2 API specification

A high-level overview of the reputation interfaces is provided in Table 10.

Table 10 - Technologies in the reputation system and policy manager

Interface /Topic	Technology and information items
Attestation Cue	<ul style="list-style-type: none"> • CBOR (to comply with RATS specifications) • Information Items to be defined
Reputation Updates	<ul style="list-style-type: none"> • JSON format, and exchanged when there are modifications in the reputation • Information Items: <pre>{ currentScore: <value of Reputation between 0 and 1>, previousScore: <value of Reputation between 0 and 1>, entityID: <AloT Identifier>, infoSIM: <String> action: <ID of action 1-deny, 2-throttle, 3-accept> actionRatio: <percentage number> }</pre>
Policies Updates	<ul style="list-style-type: none"> • JSON format, ad exchanged when there are CRUD operations in the policies • Information Items: <ul style="list-style-type: none"> ○ aiotIDs: An array with AloT identifiers; ○ infoSIM: information to SIM;

	<ul style="list-style-type: none"> ○ action: deny (1), throttle (2) and accept (3); ○ actionRatio: integer with the ratio to reduce bandwidth; ○ minReputationScore: float value representing the minimum reputation score; <p>maxReputationScore: float value representing the maximum reputation score.</p>
--	--

3.3.4 Evaluation and results

This section documents the research results regarding the alpha beta testing distribution.

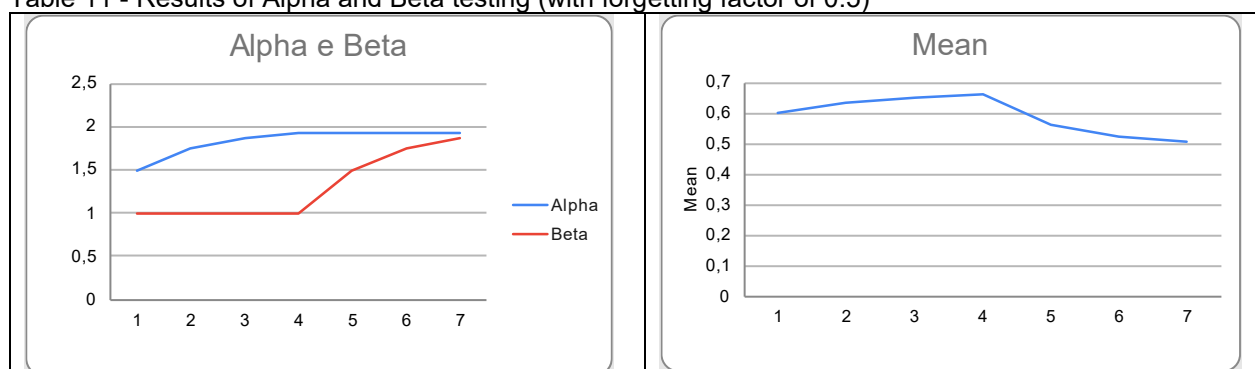
3.3.4.1 Experiment with Alpha-Beta model

A small script was made in Java to verify if the model is adequate to calculate and update the reputation value to test the Alpha Beta model. With this, several tests were conducted where the value of the ageing factor was varied, a low, medium and high value, to assess the influence of this factor in the reputation calculation. Next, random events (positive and negative) were created to determine the alpha and beta values trend. Finally, with this information, the results were added to a CSV file to be able to evaluate the alpha, beta, variance and mean values.

The tests done to test the model were as follows:

- Ten positive events with an ageing factor of 0.5
- Ten positive events with an ageing factor of 0.2
- Ten positive events with an ageing factor of 0.8
- Ten negative events with an ageing factor of 0.5
- Ten negative events with an ageing factor of 0.2
- Ten negative events with an ageing factor of 0.8
- Four positive and three negative events with an ageing factor of 0.5
- 20 random events with an ageing factor of 0.5

Table 11 - Results of Alpha and Beta testing (with forgetting factor of 0.5)



As per the achieved results there are different values that can be employed for the real value of the reputation score: mean, variance. The results demonstrate that using the mean it is possible to capture the impact of positive and negative events in the overall reputation score. For instance, from the event 4 the beta value increases and the mean value of the reputation decreases.

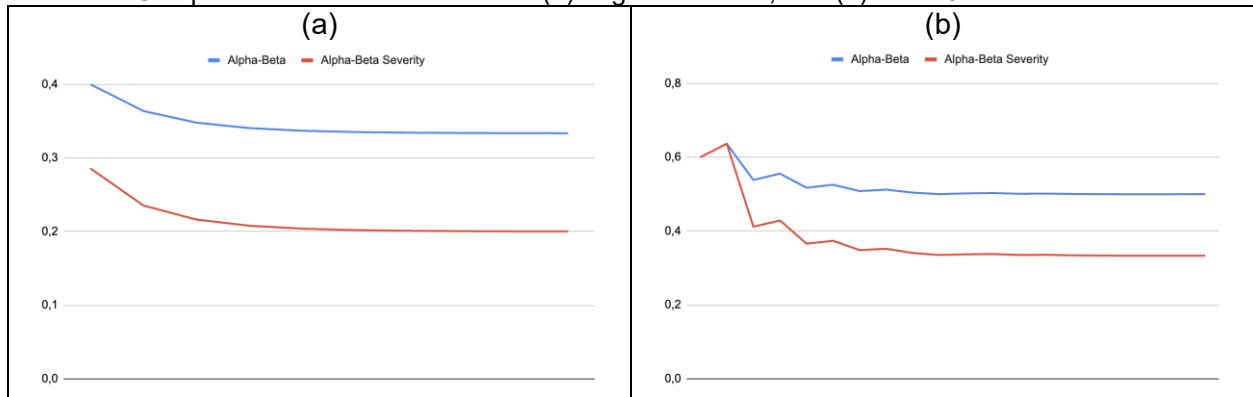
3.3.4.1 Experiment with Different reputation models

In the context of the second series of examinations, conducted to assess the impact of distinct reputation models, specifically the Alpha-Beta model and the Alpha-Beta model incorporating

Severity considerations, a series of tests were executed. The aim was to discern the fluctuations in reputation across the various models under scrutiny.

As illustrated in Table 12, it is discernible that the severity model imposes a markedly augmented penalty concerning negative events. In the specific context of the tests conducted, events designated as negative with a severity factor of 2 were considered.

Table 12 - Comparison between models with (a) negative events, and (b) with 20 random events



The instance portrayed in Table 12 offers an illustrative case wherein half of the events have a positive connotation while the remaining half embodies a negative one. By the Alpha-Beta model's anticipatory outcome, such an equilibrium would typically culminate in a reputation value of 0.5. Nonetheless, integrating a severity factor introduces a significant divergence, whereby negative events acquire a substantially amplified influence compared to their positive counterparts. Consequently, the reputation derived from a severity-modulated perspective is less than 0.5.

Another evaluation scenario considered a sequence of events consisting of 4 positive occurrences followed by two negative incidents originating from a specific entity. It is noteworthy that the negative occurrences have different severity values. As seen in Figure 59, it becomes evident that the system's imposition of penalties against the Alpha-Beta model incorporating Severity is significantly augmented. Furthermore, in alignment with the analysis of trustworthy entities, and under the premise that entities boasting reputation scores surpassing 0.5 are deemed trustworthy, while those below 0.5 are classified as untrustworthy, the entity subjected to reputation assessment via the Alpha-Beta model attains a status of trustworthiness (final reputation score of 0.5254). In stark contrast, the other two entities that used the Alpha-Beta model with Severity garnered an untrustworthy reputation status (final reputation score of 0.3735 with severity 2 and 0.2897 with severity 3).

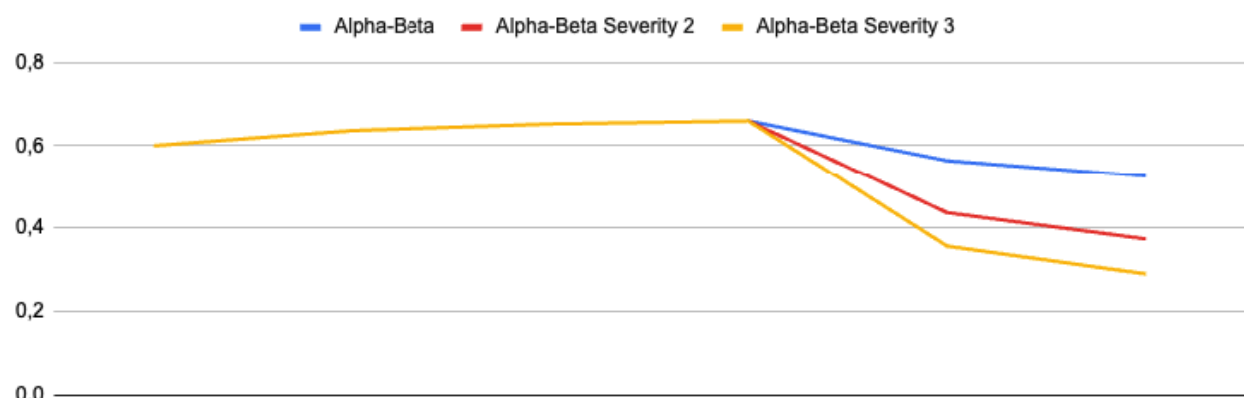


Figure 59 - Comparison of reputation score with different severity factor

3.3.5 Future work

The upcoming activities will focus on the integration of reputation system with the permissioned blockchain in the context of WP5.

The employment of reputation information will also be researched in other areas. For instance, to allow the mitigation of attacks in VoIP systems.

3.4 Remote Attestation

3.4.1 Overview

3.4.1.1 Description

The Remote Attestation solution for ARCADIAN-IoT (RA2IoT) aims to ensure that the application/services, IoT devices, and the data required for their functioning (e.g., configuration information) have not been modified and can be considered trustworthy. The solution is designed to cope with the heterogeneity regarding devices' capabilities (particularly the absence of hardware security mechanisms), leveraging Secure Elements (eSIM, or crypto chip⁶¹) as Root of Trust, thus supporting both constrained IoT devices (e.g. drones, industrial devices) and less constrained ones like smartphones.

The remote attestation procedure implements challenge-response mechanisms and is aligned with the IETF Remote Attestation Procedures (RATS) working group [54] (e.g. considering recommended formats for describing claims and associated evidence, or procedures to deliver these claims). Additionally, approaches for remote integrity assessment of the claim measurement processes have been researched.

Finally, the role of remote attestation (i.e. generated attestation results) for modelling the trustworthiness of both IoT devices and services (via remote attestation of IoT devices directly managed by / under control of the IoT service provider) has been explored (*in tandem* with Reputation System research work).

3.4.1.2 Requirements

The following requirements have been specified in D2.4:

- Requirement 5.4.1 - **Attestation pre-installation**: The (IoT) device must have or enable the Attestation component pre-installation to enable Remote Attestation procedures.
- Requirement 5.4.2 – **Serialization**: A common serialization format should be used for both Evidence and Attestation Results, to minimize code footprint and attack surface area.
- Requirement 5.4.3 – **Watchdog timer** - A watchdog timer should be implemented in a environment with some level of protection to enable receiving regular and up-to-date Attestation Results.
- Requirement 5.4.4 – **Protocol data integrity** - The integrity of Evidence and Attestation Results should be protected (i.e., either via signing or a secure channel).
- Requirement 5.4.5 – **Attestation procedure confidentiality** - Confidentiality of Evidence and Attestation Results should be protected via encryption.

3.4.1.3 Objectives and KPIs

The work to be pursued for the remote attestation system has been decomposed in two key objectives:

- Supporting Remote and Functional Attestation providing Root of Trust mechanisms with Secure Elements
- Supporting Remote Attestation involving multiple verifiers

In the following, we present the KPIs associated with the two key objectives above.

KPI scope

⁶¹ The support of Remote Attestation in industrial devices is partially supported by BOX2M, which implements the Attester functionality. Unless explicitly mentioned, the current section refers to Remote Attestation supported by Hardened Encryption component where eSIM acts -based RoT.

Novel RATS-based Remote Attestation	
Measurable Indicator	
Number of devices/OS platforms supported by remote attestation	
Target value (M30)	Current value (M20)
At least 2	2 – Android, Linux-based devices

KPI scope	
Attribute-Based Encryption for Evidence	
Measurable Indicator	
Usage of Attribute-based encryption in evidence encryption	
Target value (M30)	Current value (M20)
Used	Used

KPI scope	
Secure Element (SE)-based hybrid RoT for RA	
Measurable Indicator	
Supported secure elements (eSIM or cryptochip) as Root of Trust	
Target value (M30)	Current value (M20)
At least 1	2 – support of eSIM via Hardened Encryption (in IoT devices and smartphones), and by cryptochip (in highly constrained industrial devices)

KPI scope	
Watchdog-based attestation triggering at Verifier	
Measurable Indicator	
Availability of a watchdog-based functionality to make sure the device is periodically attested	
Target value (M30)	Current value (M20)
Available	Available

The following additional KPIs have been identified during the project as a result from the design and architectural work, where the role of attestation towards reputation modelling was identified:

KPI scope	
Support of Attestation Cues from Reputation System (for initiating new Remote Attestation processes)	
Measurable Indicator	
Availability	
Target value (M30)	Current value (M20)
Available	Available

KPI scope	
Attestation Results feeding both device and service reputation models	
Measurable Indicator	
1. Types of IoT devices reputation affected by Attestation Results 2. Number of IoT services reputation affected by Attestation Results	
Target value (M30)	Current value (M20)
1. At least 1 2. At least 1 (among domains A, B and C)	1. 2 – Smartphone (Android), Linux-based IoT device 2. 3 – All ARCADIAN-IoT services are supported

3.4.2 Technology research

3.4.2.1 Background

3.4.2.1.1 OWASP IoT Top 10 and Remote Attestation

OWASP IoT Top 10 is an online publication that gives insights into the security loopholes present in the system, and results from a thorough review of the existing cybersecurity panorama. The 10 main threats as of 2018⁶² are depicted in Figure 60. The Remote Attestation solution under specification is expected to address several items of the following list:

1. **Insecure network services:** one of the approaches suggested in the online report is to ensure the installation of regular updates. By appraising software or firmware versions as evidence, the Remote Attestation solution enables service providers to ensure only updated devices will be authorized to access IoT services.
2. **Insecure ecosystem interfaces:** attestation goes beyond IoT endpoint authentication and enables service providers to define policies which endpoints must cope with continuously for being considered trustworthy.
3. **Use of insecure or outdated components:** As mentioned in item 1., remote attestation is used to appraise software or firmware versions as evidence. Besides this, it is also used to analyse hardware models and versions, aiding service providers to enable access only to IoT endpoints which possess hardware and firmware known as secure.
4. **Insufficient privacy protection:** ARCADIAN-IoT's Remote Attestation ensures devices' evidence confidentiality via Hardened Encryption (itself supported by eSIM or cryptochip as RoT to sign encrypted evidence to ensure its integrity and trust).
5. **Lack of device management:** Remote Attestation is leveraged by the Reputation System (as its Relying Party). The Reputation System, upon the specific policies of the service provider (and its domain), will compute trustworthiness/reputation of IoT devices based on attestation results (and other information), affecting their authorization levels (e.g. leading to blacklisting).

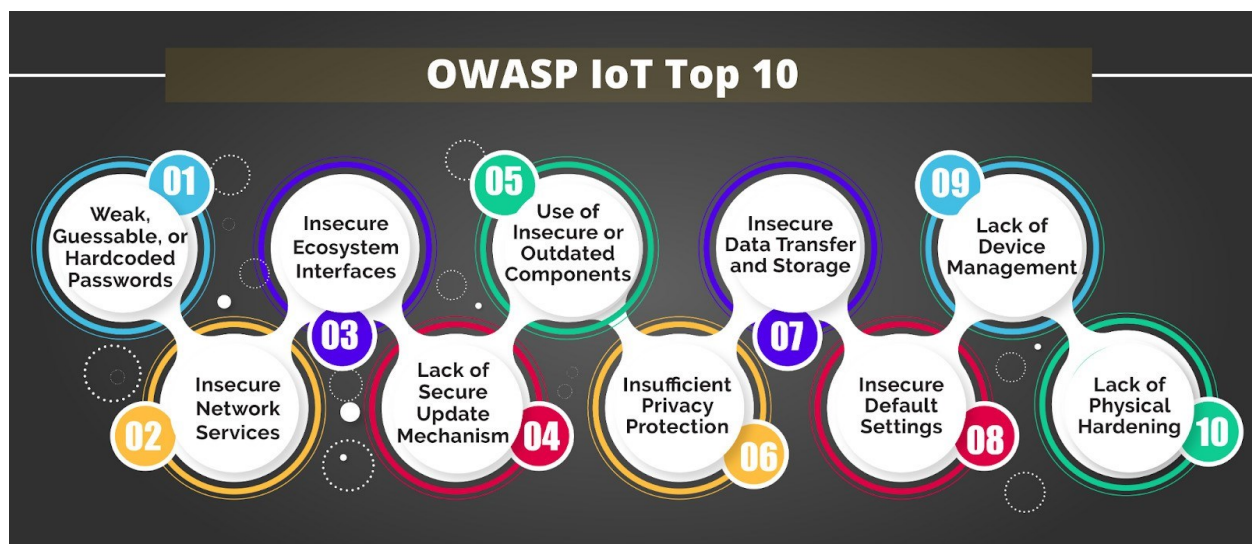


Figure 60 - OWASP's top 10 IoT security issues (2018 version)

⁶² OWASP IoT Top 10. Link:
https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10

3.4.2.1.2 Evidence format

CBOR is one of the data models considered within IETF RATS WG. CBOR is a binary data serialization based on JSON data model that is designed for small code size and small message size, such as encryption keys, graphic data, sensor values, among others. It is defined in IETF RFC 8949 [54], and was driven by the specific needs of IoT³, where devices have limited capabilities and are supposed to run with low power. CBOR is encoded in (machine-friendly) binary, instead of (human-oriented) JWTs (human-readable) JSON, importantly saving in bulk data and allowing faster processing. This data format is the recommended data serialization layer for the CoAP Internet of Things protocol⁶³, which is used for instance by CHARRA implementation of the IETF RATS model for Remote Attestation procedures.⁶⁴

A CBOR data item is encoded to or decoded from a string carrying a header byte containing a 3-bit type and 5-bit short count. This is followed by an optional extended count and an optional payload. Because of its efficiency, practicality, and support of key protocols, it was decided to use this format to encode the claims.

3.4.2.1.3 Adversarial Assumptions

An adversary's goal is to compromise a device without being detected by the Verifier. A recent survey in the scope of remote attestation for embedded devices describes a hierarchical list of adversarial assumptions regarding the level of access of the attacker to the attester:

- **Remote Adversary:** the attacker can launch remote attacks against the attester.
- **Local Adversary:** the attacker is on the same network as the attester and can interfere with its communications.
- **Physically Non-Intrusive Adversary:** the attacker is physically close to the device but not able to interrupt its service; side-channel attacks (including secrets extraction) against the attester are possible but not power-related or physical tampering.
- **Physically Intrusive Adversary:** the attacker is in possession of the attester and can power it off and physically tamper with its hardware.

Being hierarchical, any adversary in a given level of the hierarchy is capable of performing all the attacks from adversaries at lower levels of the hierarchy, with Physically Intrusive Adversary being at the top level and Remote Adversary at the bottom. Solutions are however not hierarchical themselves, as they may be focused at particular issues or types of attacks. In other words, the malware model clarifies what type of malware will be defended against by the RA scheme and **measures the assumed maximum level of compromise that a RA scheme will defend against.**

Scenarios where the adversary is in full control of the attester's network are out of scope of the ARCADIAN-IoT's Remote Attestation solution.

The same survey further extends the threat model to include a malware model measuring the **ability of a dishonest/compromised attester to subvert attestation**. The identified threats (i.e., Service File Malware, Service File-less Malware, Device file Malware, Device File-less Malware) vary by target (i.e., attester's service or the device itself) and where they will reside (i.e., in the storage or in the RAM), which impact the associated danger/impact and footprint (evidence of compromise).

3.4.2.1.4 Types of Attacks

Several types of attacks to remote attestation solutions have been documented in the literature. Some specifically target RA solutions which use timing as root of trust, where the Verifier knows the precise latency of the attester's evidence collection process as well as the round-trip-time of the attestation request. Attacks include:

- Proxy attacks

⁶³ <https://en.wikipedia.org/wiki/CBOR>

⁶⁴ <https://github.com/Fraunhofer-SIT/charra>

- Precomputation
- Overclocking
- Evidence gathering optimization

While those attacks can be defended via adequate security strategies (e.g. random challenge time), timing-based RoT's main limitations are the application in real settings, where timing can be affected by varying network communication conditions such as congestion and the resulting jitter. There are also attacks specifically targeting RA approaches applying discrete evidence collection:

- Memory copy (file)
- Memory copy (fileless)
- Compression (of malware, legitimate files, etc)
- Split Translation Lookaside Buffer (TLB)
- I-cache inconsistency
- Time of Check Time of Use (TOCTOU, where an attacker is able to compromise an attester and removing associated evidence before attestation time)
- Return Oriented Programming (ROP)
- Data Oriented Programming (DOP)

Some guidelines for mitigating attacks against discrete evidence collection include applying randomness in the time interval between the moments where the challenge is sent and in the memory walk strategy, or writing evidence gathering compilation code in a way that doesn't allow compression or optimization.

3.4.2.1.5 Root of Trust

For the Verifier to trust the Evidence provided by the Attester, a Root of Trust (RoT) is mandatory. Without a RoT, an attester could forge evidence or provide evidence that was generated by another device or network entity. The RoT is defined by which components of the attester (hardware, software or both) are used to gather evidence. Previous software-based RA schemes provide assurance over the device (and its evidence) by one of three main options, described below:

- Virtualization using a hypervisor:
 - The major drawback of this option is that, not all devices may support the additional computation and latency. This is particularly true in the case of IoT devices.
- Filling excess memory with random noise:
 - Here, the level of noise is compared by the Verifier against a Reference value, as any modification to the program memory will force the system to erase some of the noise. In case the device is compromised, its attacker will not be able to recover the same level of noise, being as such detected due to attestation failure.
 - The major drawback is that, modern embedded devices have processes which may change memory while operating normally, which would cause the attestation to fail; compressing programs for an embedded system is not trivial.
- Timing & timestamping:
 - In this approach, the verifier compares the latency of the attester's evidence collection algorithm and the exact round trip time of the attestation request with a reference acceptable timeframe.
 - However, due to network jitter in real environments it is not possible to rely on such approach.

The *hybrid root of trust* is typically defined as using a combination of hardware and software features; more concretely, it can be narrowed down to any RoT that uses specific hardware features that are already available on certain embedded devices (e.g. TEEs such as the Intel SGX or the Trusted Computer Group's TPM, Memory Protection Units (MPUs), write-protected clocks, and ROM). *Hardware RoT* is considered as any purpose-built hardware that gathers and provides evidence to the verifier for RA (e.g. a redesigned processor, security co-processors, or accelerator in the system).

ARCADIAN-IoT builds on a hybrid RoT, supported by Hardened Encryption and eSIM components. It was identified the such approach does not provide a Root of Trust for claims measurements.

3.4.2.1.6 *Nature of Claims/Evidence*

With respect to their nature, two major types of evidence exist: Static or Dynamic. **Static evidence** refers to evidence that does not change over time, such as boot sector of memory, executable binary files, software configurations or information about hardware components. This type of evidence enables the detection of malware attacks, which reside in the disk/storage of the device (i.e., Service File Malware, Device File Malware), such as settings or binary modification. However, it does not enable detection of changes in memory-resident resources (e.g., kernel or system call table).

Dynamic evidence refers then to evidence that may change over time, such as RAM contents, information about running processes, contents of instruction cache (I-cache) or data cache (D-cache). It enables in principle the detection of any type of malware, being it resident in the storage or on the memory. Some dynamic evidence, such as cache contents, is more difficult to collect and require hardware root of trust.

3.4.2.1.7 *Claim collection methods*

Discrete RA schemes collect evidence at a particular moment in time and are well suited for static evidence. **Continuous RA** schemes collect evidence from the attester device over a period of time. Continuous RA schemes are usually implemented along with a static RA. In general, these schemes involve continuously/periodically monitoring the behaviour of given component on an attester device.

3.4.2.2 *Research findings and achievements*

3.4.2.2.1 *Remote Attestation for IoT devices and services*

A novel Remote Attestation scheme for IoT devices and services, aligned with the IETF's RATS group specifications, has been researched and implemented. It implements a challenge-response model, via Remote Procedure Calls (RPC), where the Verifier invokes a remote procedure on the Attester, with the attestation request as its parameters. The Attestation Request (sent to the Attester), the Attestation Response (sent to the Verifier), and Attestation Results (sent to the Reputation System) are encoded in CBOR format, in line with IETF RATS guidelines. The main achievements can be highlighted:

- a) Implementation of a complete Remote Attestation solution leveraging a hybrid Root of Trust (RoT) for ensuring the trustworthiness of the encryption mechanism and provenance of the evidence sent by the attester. The hybrid Root of Trust relies on
 - a. a distributed, multi-authority Attribute-based Encryption (ABE) to protect evidence data confidentiality and enabling different Verifiers to access different sets of claims
 - b. eSIM, a technology increasingly present in IoT devices whose main role is related to mobile and secure connectivity-provisioning, to sign the attestation evidence;
 - c. software-based processes for assessing claim measurements integrity (described further below)
- b) Implementation of a Verifier component tailored for supporting remote attestation of highly constrained devices, via the appraisal of evidence signed and encrypted by a cryptochip (BOX2M), acting as RoT.
- c) Support for two remote attestation initiation methods: periodically, in a "watchdog-based" approach, where an attestation challenge is sent whenever a certain amount of time has

passed since the last attestation procedure; and via an “attestation cue” from the reputation system, according to its policies.

3.4.2.2.2 Contribution to IoT devices and services reputation

The attestation result, whose contents and meaning were established according to its usefulness for trustworthiness modelling (via the Reputation System), contains two values between 0 and 1, each representing the proportion of valid claims and an integrity score (described in the section below). The Reputation System acts as the Relying Party with respect to the Remote Attestation procedure, with the Remote Attestation of IoT devices representing a crucial source of information leverage by the Reputation System for modelling Trust of both devices and associated services, as follows:

- The attestation of user controlled IoT devices contributes to the trust modelling of IoT devices
- The attestation of service controlled IoT devices contributes to the trust modelling of both the devices and respective services which they support

3.4.2.2.3 Remote integrity assessment of processes running in IoT devices

As mentioned previously, the solution relies on a hybrid RoT, where a combination of eSIM and Attribute-based Encryption is used to ensure integrity of evidence signature and encryption, respectively. However, considering that the eSIM only ensures signature integrity and the provenance of the evidence / set of claims, this leaves a security gap, namely with respect to the integrity of the measurements provided by the device being attested. This leads to the question about whether the Verifier trust that the received measurements have not been tampered with before being encrypted and sent by the target Attester. With this issue identified, our next research steps were guided by the question: *How can we devise a way to trust that the claims reported by the device are true?*

We consequently researched a software-based RoT approach for improving the confidence in the integrity of the claim measurements. As design goals, we established that:

- the RoT approach should be as much device-agnostic as possible
- adaptable
- scalable
- computationally efficient, for being coping with energy- and/or computationally-constrained devices.

Based on the above design goals, particularly the computational-efficient requirement, it was identified that most of the computation should be performed on the Verifier side (where resource consumption is less of a problem), leaving only the collection of the data (sets of claims) to the Attester. Considering then that the evidence processing is prevalently processed remotely, it is concluded that the associated analysis must rely on dynamic, incomplete information; under such assumption, at the most fundamental level, the targeted approach must learn a distribution of the behaviour or a baseline for some “characteristics” of the Attester’s trusted state and detect deviations — i.e., anomalies — from that. This reasoning led to the conclusion that an AI/ML-based approach for assessing the integrity of the claims collection / measurements would be adequate.

The next step was to define a target — i.e., type of attacks — to detect, and associated threat assumptions. It was decided to address Code Reuse Attacks (CRAs). In short, CRAs are a type of attack where the attacker seeks to exploit vulnerabilities in the existing code on the device, hijacking its control and/or data flow, by sending well-crafted input to the device, granting, for example, root access to the device or the ability to execute arbitrary code.

The choice of CRA attacks was essentially driven from the fact that in a typical IoT device it may not be possible to execute new software or firmware, in which cases attackers might rely on other

approaches to exploit these devices. CRA attacks are a relevant type of attack where the code already running on the device is exploited.

In summary, to assess the trustworthiness of the claims provided by an attester, an approach would be to detect the probability of processes (i.e., programs) on the device being subject to CRAs. We note that this is an indirect approach, since the trustworthiness of the claims are not directly assessed; instead, we determine the probability of the existence of CRA processes on the device. It should be noted that detecting CRAs is just the first step into assessing the integrity of the device, in the future the detection of other types of attacks might be included.

The general process is the following. On the Attester, execution traces of specific processes, representing the order of instructions being executed in the target processes, are collected and sent to the Verifier. The Verifier, a Deep Learning model, analyses these traces and assesses their deviation from some baseline, resulting in a probability of the existence of a CRA. This probability is referred to as the Integrity Score and is included in the attestation result sent to the Reputation System, influencing the Trust score of this device. The work associated to this mechanism is currently still at preliminary research stage, with an initial experimental proof of concept under implementation (i.e., current maturity level is that of TRL3).

3.4.2.3 Produced resources

The current implementation of the Remote Attestation system (RA2IoT), including its two main software components (Attester and Verifier), as well as necessary configuration files and information, can be accessed at the project's Gitlab repository:

https://gitlab.com/arcadian_iot/remote_attestation.

3.4.3 Design specification

3.4.3.1 Logical architecture view

The novel Remote Attestation solution proposed within ARCADIAN-IoT is itself enabled by the novel combination between Hardened Encryption's Attribute-based Encryption (ABE) and GSMA IoT SAFE-compliant eSIM as RoT for signing the resulting encrypted data (i.e. the hash). In general, the eSIM signature as RoT strengthens the encryption process by ensuring data provenance and avoiding impersonation attacks where malicious agents send data on behalf of other devices. The same general principle is transposed to the Remote Attestation process, with the eSIM acting as RoT for transmission of attestation evidence data from the attested device. To obtain further guarantees regarding the trustworthiness of the measured claims, and because of the absence of a hardware-based RoT (e.g. TPM) for such measurements, RA2IoT will leverage Machine Learning for assessing the integrity of the device through the analysis of specific processes, especially those relevant to the claim collection, whose result will also be included in the attestation results.

Furthermore, the result from the attestation process is processed by the Reputation System, affecting entities reputation according to its policies (which are defined according to the service provider's or other relevant stakeholder needs).

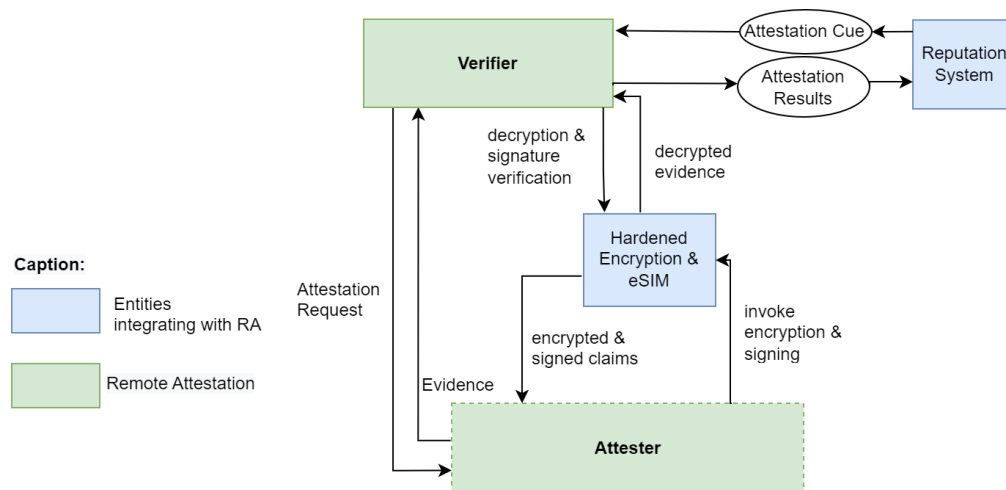


Figure 61 - Remote Attestation logical architecture and external dependencies

3.4.3.1.1 ARCADIAN-IoT Verifier

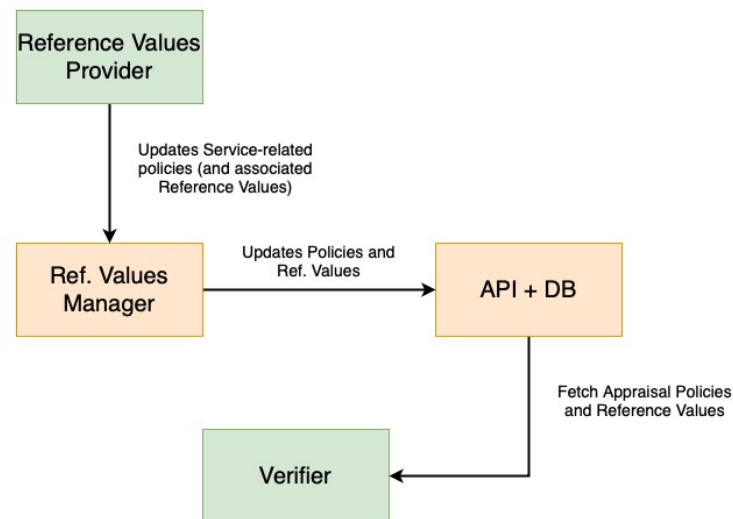


Figure 62 - Verifier's Internal logical architecture

Figure 62 shows two components that support the Verifier's internal operations (directly and indirectly). The database, accessed via an API, is used to store all the data required for the Verifier's operations, such as the claim selection to be sent in the attestation request or the appraisal policies and reference values for the appraisal of evidence. The other component, Reference Values Manager, receives updates about the Reference Values and associated policies from the Reference Values Provider (e.g., a Service Provider) and updates the database accordingly, also via an API.

3.4.3.2 Sub-use cases.

Reference Value transmission for Device Attestation

In this case, acceptable reference configuration values for attesting the claims associated to device integrity are obtained by the device Verifier. These are typically sent from a Reference Value provider (e.g., the manufacturer).

Reference Value transmission for Service Attestation: Service Provider sends acceptable reference configuration values to the Verifier responsible for attesting the service-specific claims.

Remote Attestation Procedure for Devices

Remote Attestation procedures can be initiated according to two different ways:

- Via an Attestation Cue from the Reputation System (e.g. upon sudden reputation decrease, event received from Behaviour Monitoring)
- Regular, watchdog-based attestation cycle, initiated by the responsible Verifier (which may either be responsible for appraising device, service-specific or both types of evidence).

Remote attestation is used to appraise devices' evidence, which includes both a nonce and a set of claims (e.g. HW model, build version) and assess the existence of possible compromises to the device's integrity. The attestation result generated via evidence appraisal represents a percentage of the claims that are valid according to the appraisal policies, or 0 in case of critical issues with the attestation response (e.g., wrong nonce), and the level (percentage) of trustworthiness on the device. These results are sent to the Reputation System for interpretation and to adjust the device's reputation value.

Each of the previously mentioned reference use cases can also be distinguished according to the type of devices (i.e. smartphone, industrial IoT device, drone), where the reference values and appraised evidence will be different.

Given the absence of a HW-based Root of Trust for the claims measurement process, the Remote Attestation procedure also embeds a software-based integrity assessment feature, aiming to provide further guarantees over the integrity of the provided / measured claims (described in section 3.4.2.2.33.4.2.2).

3.4.3.3 Sequence diagrams

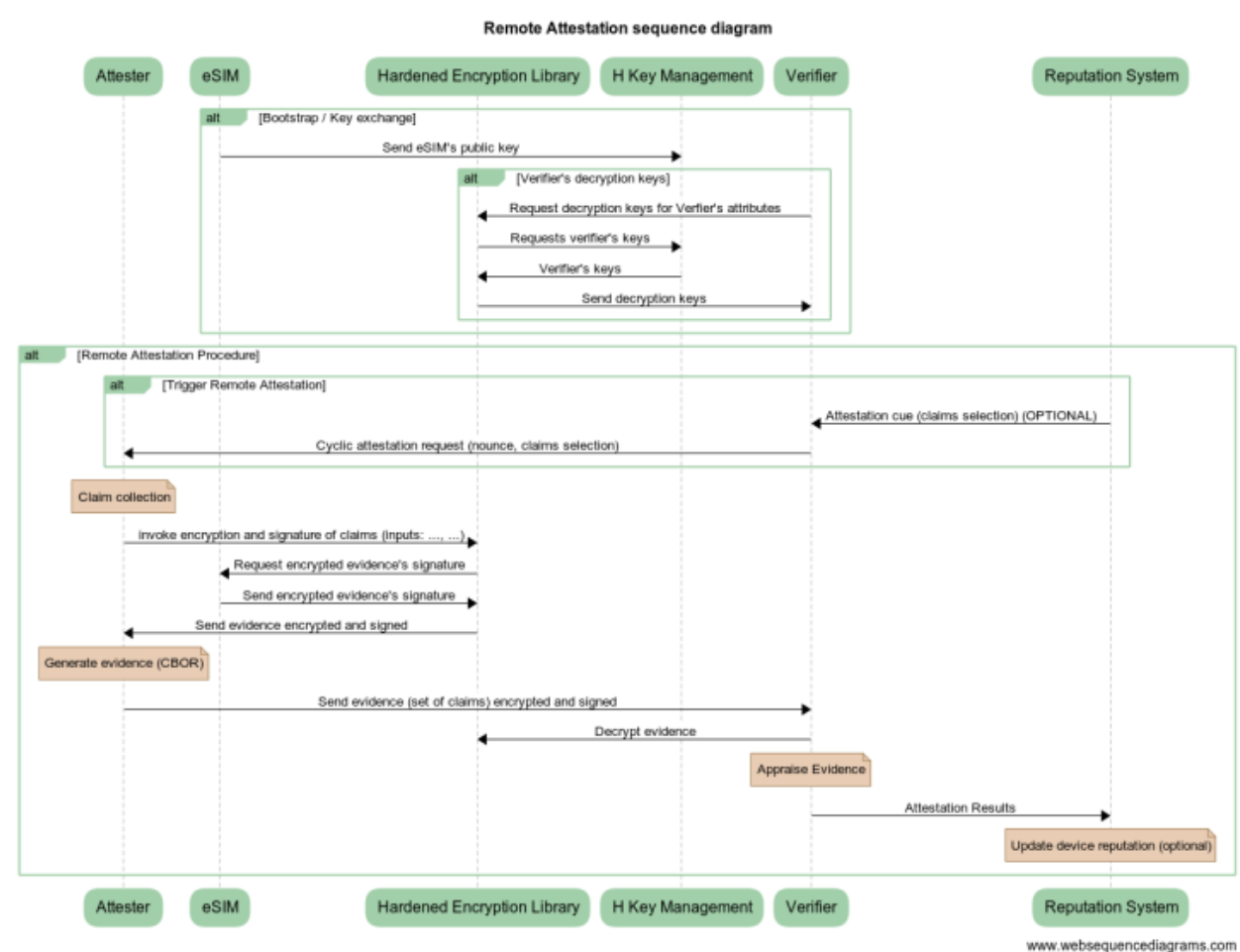


Figure 63 - Remote attestation bootstrap & execution procedure signaling

Note: While a single Hardened Encryption Library box is represented, Attester and Verifier leverage distinct Hardened Encryption libraries instances: the former at the device, and the latter at the infrastructure / network side.

3.4.3.4 Interface description

3.4.3.4.1 Internal interfaces

Attester – Verifier

This interface is used by the Attester to send the evidence to the Verifier. Reversely, the Verifier uses it to trigger remote attestation procedures. A previous implementation used the Constrained Application Protocol (CoAP) for evidence transmission from the Attester to the Verifier. CoAP is specified by RFC7252 and is a document transfer protocol like Hypertext Transfer Protocol (HTTP). It has been designed from scratch for constrained devices, leveraging bit fields and mappings to keep the packets as small as possible. CoAP is based on a simple client/server model and follows the RESTful paradigm. Moreover, CoAP uses User Datagram Protocol (UDP) as transport protocol; alternatively, the secure Constrained Application Protocol (CoAPs) scheme utilizes DTLS (instead of UDP) and guarantees confidentiality, integrity and authenticity of the CoAP packets. However, some limitations in using CoAP for the Attester – Verifier communication were identified, namely scalability, since the verifier must be blocked. As such, as a step towards a more scalable solution, it was decided to implement the communication using RPC via RabbitMQ. As of now, this approach follows the same challenge-response model as was previously done via CoAP, where RPC is used to send the attestation request and receive the attestation response. The goal, as future work, is to evolve this process to an event-stream (subscription) base model where RPC is used by the Verifier(s) to subscribe to evidence updates from an Attester, which in turn sends periodical updates. This will allow Verifiers to manage the attestation of a wider, constantly changing range of devices more efficiently.

Every time the Attester sends the attestation response, a nonce is included to minimize threats against (and enhance trustworthiness in) the attestation procedure. A nonce is a random sequence (e.g., of bytes) that uniquely identifies each Attestation Request. Its purpose is to ensure the freshness of information and prevent replay attacks. This random sequence is generated by the verifier and sent to the attester in the attestation request. Upon receiving it, the attester must copy it and include it in the evidence, along with the claims. Claims and nonce, forming the evidence, are then encrypted and sent to the verifier, that checks if the received nonce is the one expected before evaluating the claims. If the nonce is different, then the attestation process fails.

The following interfaces are internal to the Verifier operation:

Reference Values Manager – Database

A Reference Values Manager component is responsible for managing data such as Reference Values and associated Appraisal Policies, according with the information provided by Reference Value Providers, such as Service Providers and other sources of information.

Verifier – Database

To generate the claim selection to send on the Attestation Challenge and to access the Reference Values and respective policies for appraisal of Evidence, the Verifier access a Database, kept up to date by the Reference Values Manager.

3.4.3.4.2 External interfaces:

Interfaces used in scope of eSIM-based Hardened Encryption

Attester – Hardened Encryption (libraries)

The Hardened Encryption libraries are used by the Attester to both perform cryptographic operations and to retrieve its ABE keys. It begins with a registering step, where it uses the provided libraries to request its keys, according with its attributes to the Hardened Encryption - Key Management sub-component. When, upon request by a Verifier, it generates evidence, the libraries are used to encrypt and sign (with its eSIM key) this evidence, before sending it to the verifier.

Attester – Hardened Encryption (Key Management)

The Attester obtains encryption keys from the Hardened Encryption (Key Management) during the initial registration step, for each of its attributes, that are used for its cryptographic operations.

Verifier – Hardened Encryption (Key Management)

The Verifier uses this sub-component in the same way as the Attester's.

Verifier – Hardened Encryption (library)

In general, the Verifier uses this library in the same way as the Attester. It begins with the registering step but then, instead of encrypting, it decrypts the evidence received from the Attester.

Interfaces used in scope of cryptochip-based Hardened Encryption**Verifier – IoT Middleware**

The Verifier uses an API provided by the IoT Middleware (owned by the service provider) in order to initiate the Attestation process by sending an HTTP request containing the device ID and a nonce. Upon receiving the request, the Middleware is responsible for gathering data/claims from the device and send it back to the Verifier. The security and integrity of the claims is ensured by the cryptochip-based Hardened Encryption (from the device to the Middleware) and via the HTTPS protocol (from the Middleware to the Verifier).

Interfaces used in in both cryptochip-based and eSIM-based Hardened Encryption

For both types of HW-based encryption, additional information exchange with external entities is done via the ARCADIAN-IoT message bus exchange *ra_exchange*, namely:

Verifier – Reputation System

The Verifier transmits the attestation result (i.e. the outcome of the evidence appraisal) to the Reputation System via the ARCADIAN-IoT message bus, more specifically using *ra_exchange.attest_results* topic. The message bus is also used by the Reputation System to send Attestation Cues (for initiating remote attestation procedures upon its policies), using the *ra_exchange.attest_cue* topic;

Reference Values Manager – Service Provider

Interface used (by a Reference Values Manager) to collect Reference Values associated to ARCADIAN-IoT compliant services (e.g. DGA, Medical IoT), and to update the Database shared with the Verifier. Similarly to the interface with the Reputation System, the information will be received via the message bus, using the *ra_exchange.ref_values* in this case.

3.4.3.5 Technical solution

3.4.3.5.1 Deployment architecture view

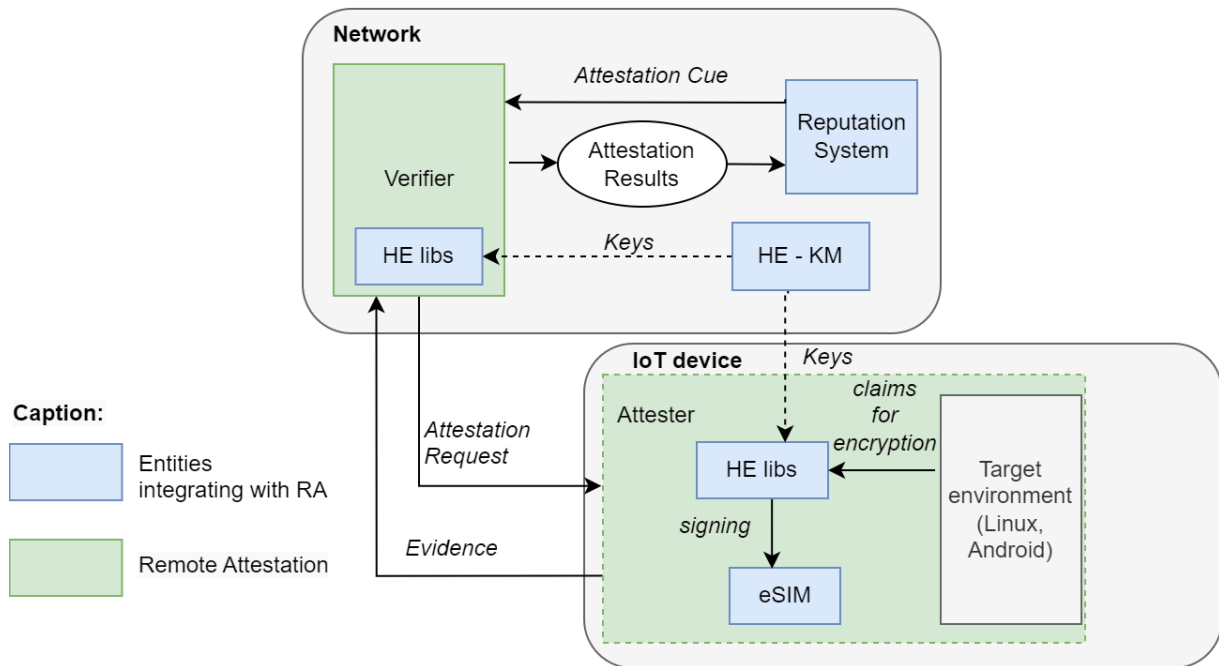


Figure 64 - Remote Attestation for ARCADIAN-IoT (RA2IoT) deployment architecture view

3.4.3.5.2 API specification

No APIs are provided by Remote Attestation. Information exchange with external entities is done via the ARCADIAN-IoT message bus exchange *ra_exchange*.

3.4.3.5.3 Security aspects

IETF's draft on Reference Interaction Models for Remote Attestation Procedures⁶⁵ states two essential requirements to be fulfilled so that the appropriate transmission of evidence is ensured. These are:

- Integrity: the information transmitted by the attester must be integral – i.e., the transmitted information should not be inadvertently modified in *any* situation;
- Authenticity: the guarantee that the information transmitted is from the *attester* that it is supposed to be.

These requirements are necessary conditions to guarantee to the *verifier* that the *attester* is the one expected and that the information was generated by this *attester*. However, they are not enough to trust the actual information – the evidence – provided by the attester and to grant permissions for it to access the specific services it might request.

To assess the trustworthiness of the evidence, first the verifier needs secure statements that provide assurance of the attester's capabilities to securely generate and transmit evidence (*endorsements*), provided by trusted entities (*endorsers*).

The main endorser is the ARCADIAN-IoT's hardened encryption component. Since the attester uses this component's library to encrypt the evidence, the endorsement takes place by having the hardened encryption's key management subcomponent provide keys to the verifier, so that it can (functionally) decrypt the evidence. Additionally, since it performs the signature of evidence, via the attester's eSIM, it ensures non-repudiation to the verifier. In more detail, the Hardened

⁶⁵ H. Birkholz, W. Pan, E. Volt, "Reference Interaction Models for Remote Attestation Procedures", (IETF draft), September 2022. Link: <https://datatracker.ietf.org/doc/draft-ietf-rats-reference-interaction-models/06/>

Encryption – Key Management subcomponent - is expected to provide to the target Verifier(s) the attribute/decryption keys (as endorsements) required to (functionally) decrypt the necessary attestation claims. The data is decrypted only if the set of attribute keys belonging to the entity satisfies the access policy. The data is always encrypted with the same public key (technically belonging to the Attribute Authority, which check entities eligibility for attribute keys and delegates them to the respective entities).

3.4.3.6 Other technical specifications

3.4.3.6.1 Target claims

Claims are taken from the target environment and, along with the nonce, are part of the Evidence. They represent characteristics of an Attester's Target Environment. The types of claims collected vary according with service requirements, context, devices, etc.

Generally speaking, for unconstrained IoT devices such as smartphones (e.g., involved in domains A and C) the following general types of claims may be considered: claims related with device and product information, and hardware and system information.

As for constrained IoT devices powered by Linux, such as drones (e.g., in domain A), possible claims may be: OS information, hardware characteristics, operational status, and gps information (in particular in the case of drones).

With respect to claims in the context of industrial devices, however, the following claims the following claims were already agreed with the domain owners: Vendor Name, Firmware Version, Device ID, and Equipping Class

Furthermore, in the scope of **Service Attestation**, the target claims are under discussion and specification with the domain owners, with one clear target being the application fingerprint.

3.4.3.6.2 Reference Values and Appraisal Policies

Reference Values are sets of values provided by *Reference Value Providers* and used by the verifier – as reference – to assess (compare) the validity or suitability of the evidence's claims, in the verifier's evidence appraisal policy. Given the claims presented in the Section 3.4.3.6.1, we consider as *Reference Value Providers* the manufacturers of the devices (or of its components). As for Service Attestation, the *Reference Value Providers* will typically be the Service Provider. The exact values to be considered will be established later, driven both from the decision on target claims and from the specifications on the use cases implementation (in D5.3 and as a result of the work on Tasks T5.2-T5.4), where exact hardware and software to be used, as well as applicable appraisal policies in each domain / use case will be agreed.

Upon receiving the response to the attestation request, and having the contained evidence decrypted by Hardened Encryption, in general, the following procedure is executed by the Verifier in order to appraise the received evidence:

1. Checks if the nonce matches the one sent in the attestation request – a random value used to prevent replay attacks.
2. Appraises each claim, according with the reference values, given by the reference value providers.
3. Stores the attestation results in a data structure and sends them to the relying party (a role fulfilled in RA2IoT by the Reputation System).

3.4.3.6.3 Attestation Results

After the Evidence is appraised in the Verifier, the Attestation Results are generated and stored in a data structure before being sent to the Relying Party. The data structure contains the following attributes:

- aiotID of the device being attested.
- A timestamp of the moment the response is generated.
- The integrity level of the device, represented as a real value between 0 and 1.
- The ratio of valid claims – i.e. the number of claims which match the accepted state according to the appraisal process over the total number of appraised claims - represented as a real value between 0 and 1.

3.4.3.6.4 Supported approaches to initiate Remote Attestation

A Remote Attestation procedure of a given device may either be run **periodically**, i.e., being repeated after a given time has passed, or **on-demand**, requested by the Relying Party (the Reputation System in ARCADIAN-IoT). In the first case, remote attestation procedures are periodically triggered by an internal process of (one of its) associated Verifiers (e.g., ensuring attestation every t seconds) - we refer to this as “Attestation Trigger”. In the latter, remote attestation procedures are spawned by the Reputation System (RS) as the result of a given event and the RS stored policies (which should consider aspects including expected frequency, energy availability and consumption rate) - this is referred to as “Attestation Cue”.

Since the attestation process follows a challenge-response interaction model, the periodic triggering of the attestation process is employed by having the Verifier sending an attestation request to the Attester when the time is due. As such, when the RS requests a remote attestation procedure, what it is actually doing is *cueing* the Verifier to trigger the procedure, hence the name “Attestation Cue”.

3.4.4 Evaluation and results

The ARCADIAN-IoT remote attestation procedure leveraging ABE libraries for evidence encryption / decryption has been tested and validated using both dummy and real (collected from the attested devices' environment) data, for both the smartphone and drone. As for the Remote Attestation support in industrial devices, the communication between the Verifier and the middleware, via the provided API, has been successfully tested, and will be further validated in the context of domain B validation (WP5 scope).

3.4.5 Future work

The research and implementation of approaches for enhancing the remote assessment of the device's integrity and trustworthiness of the measurements it provides, as well as associated robustness and scalability (e.g. via federated approaches), have been identified as relevant upcoming research activities.

4 RECOVERY PLANE

4.1 Self-recovery

4.1.1 Overview

4.1.1.1 Description

The Self-recovery component is composed of a storage server, that exposes a REST API via HTTP/S and client-side (on-device) scripts, that allow devices interface with the storage server and store and retrieve backups. The types of data that will be stored will vary from device to device, ranging from configurations that are required for the device to operate normally, system logs, on-device application data and if necessary, data gathered by sensors.

The results of Hardened Encryption task will be used to secure the backups and also provide layered access policies to different level users, for example, device owners will be able to decrypt all backups, while system administrators will be able to decrypt system logs only. The preferred location of data encryption is on the device itself, though resource constraints may render the encryption process unfeasible. For addressing this issue, an encryption proxy is used to receive plain data, encrypt it and either return it to the device, or forward it to the storage module of the recovery component.

In cases where the devices are simple sensors without an operating system, the client-side recovery scripts can be instead run from a gathering/controller device, for example a phone that uses Bluetooth to connect to sensors.

The ability of a device to access the recovery services is verified during each device-server interaction; one of the metrics checked is its reputation score, meaning a device with compromised security will need to go through one or more processes to increase its trustworthiness score, such as credential recovery, described in section 4.2, before being allowed to either store or retrieve a backup.

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.

4.1.1.2 Requirements

Requirement 7.1.1 – Recovery mechanism

Each recovery system requires first an organised and detailed description of a running system. As a second step we need to collect all data that define a targeted system or process and all processes that are needed to set a system in an operational state.

From the resource aspect, the recovery system needs the access to

- the data required for recovery,
- a set of scripts that set up the recovery processes and
- the machine which can run the recovery process and with access to the services and/or infrastructure that require recovery (network connectivity, etc).

In addition to the requirements previously outlined in WP2, recovery scripts will also perform periodic data backups that will be uploaded to the Self-recovery data storage server. The backups will be encrypted both in transit and at rest using the results of the Hardened Encryption task.

Remote data backups will enable quick replacement of devices that are either malfunctioning, lost or stolen.

4.1.1.3 Objectives and KPIs

KPI scope	
The recovery process is successful if the application/process/device is running as expected.	
Measurable Indicator	
A device that experiences storage failure or is faulty and must be replaced by a new unit, can recover its backed-up data and resume functionality (after performing the credential recovery process)	
Target value (M30)	Current value (M34)
Recovery process works on actual device	Recovery process works on simulated device (VM)

KPI scope	
Data can be encrypted in a selective way, by applying a policy that defines which stakeholders, relying on their public keys, can decrypt partial or complete data.	
Measurable Indicator	
Backup encryption policies enable stakeholders to be granted selective access to different types of data based on a user's role	
Target value (M30)	Current value (M34)
Selective decryption based on user access level and encryption policy	General encryption of backups, only the device can decrypt its backup

4.1.2 Technology research

4.1.2.1 Background

Particular emphasis was placed on the selection of the storage technology for the Self-recovery component. Initially, XtreamFS was chosen for its simplicity and performance, but during the implementation phase, certain problems were encountered when the testing environment changed to newer versions of operating systems (specifically, CentOS 7 -> CentOS Stream 8). The choice of the storage backend then shifted to CEPH, which is more complex, but also more robust and has a better update and support lifecycle.

4.1.2.2 Research findings and achievements

XtreamFS⁶⁶ was the first choice for integration as a storage backend due to its capabilities of scalability, fault tolerance and relative ease of administration. While initial tests of the system were successful, subsequent testing in a cloud environment revealed issues with XtreamFS that made it untenable as a choice in a modern platform, mainly the use of older versions of dependencies

⁶⁶ <http://www.xtreamfs.org/>

that made it a challenge to run on more recent operating systems, while using old versions of software is also undesirable from a security perspective. Subsequently, CEPH⁶⁷ became the choice for the storage backend. While it is more complex, even a minimum deployment must consist of at least a monitor, that has an overview of the storage cluster status, and an OSD (Object Storage Daemon), its use once the initial setup is done is also quite simple.

The other technology choice question outlined in the previous period was the choice of the REST API framework. OpenAPI specification is the appropriate choice, as its self-documenting feature is a great help when integrating with other components. The first prototype version of the Self-recovery component uses a NodeJS implementation of OpenAPI v2, during development towards prototype 2, the specification moved to OpenAPI v3 and the server was rewritten in Golang.

4.1.2.3 Produced resources

The final prototype version of the Self-recovery component is available on the project's Gitlab repository⁶⁸. It includes the server component, client-side scripts for performing the recovery operations in Python and an Ansible deployment script that provisions a single-node CEPH cluster and the Self-recovery server component. The deployment can be tested with an included demo script that simulates a backup operation, device failure and data recovery.

4.1.3 Design specification

4.1.3.1 Logical architecture view

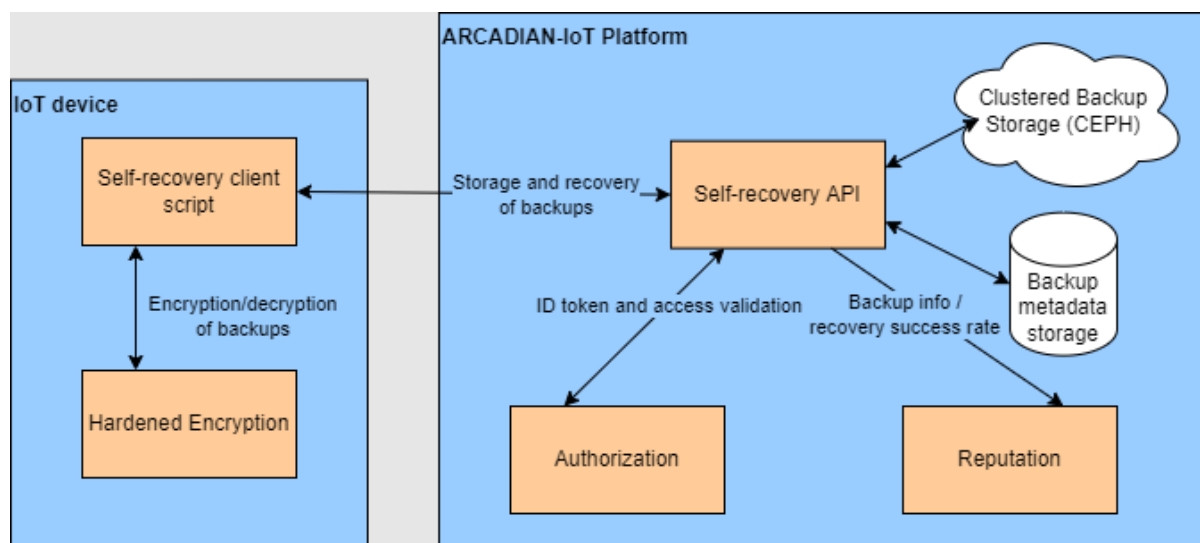


Figure 65 - Self-recovery logical architecture view

Self-recovery comprises client-side (on-device) and server modules, where the backups and related metadata are stored. Figure 65 depicts the basic structure of the component, but omits interactions with other ARCADIAN-IoT components, which will be described in the next section. The backups produced by the client-side module are sent to the server, where they are stored on a CEPH cluster, while that backup metadata (size, time, signature etc.,) are stored in a standard relational database.

⁶⁷ <https://docs.ceph.com/en/quincy/>

⁶⁸ https://gitlab.com/arcadian_iot/self-recovery/-/tree/develop

4.1.3.2 Sequence diagrams

The first sequence diagram presents a simple case, where a device administrator restores a device after it was damaged or experienced a hardware malfunction. The recovery actions are triggered manually.

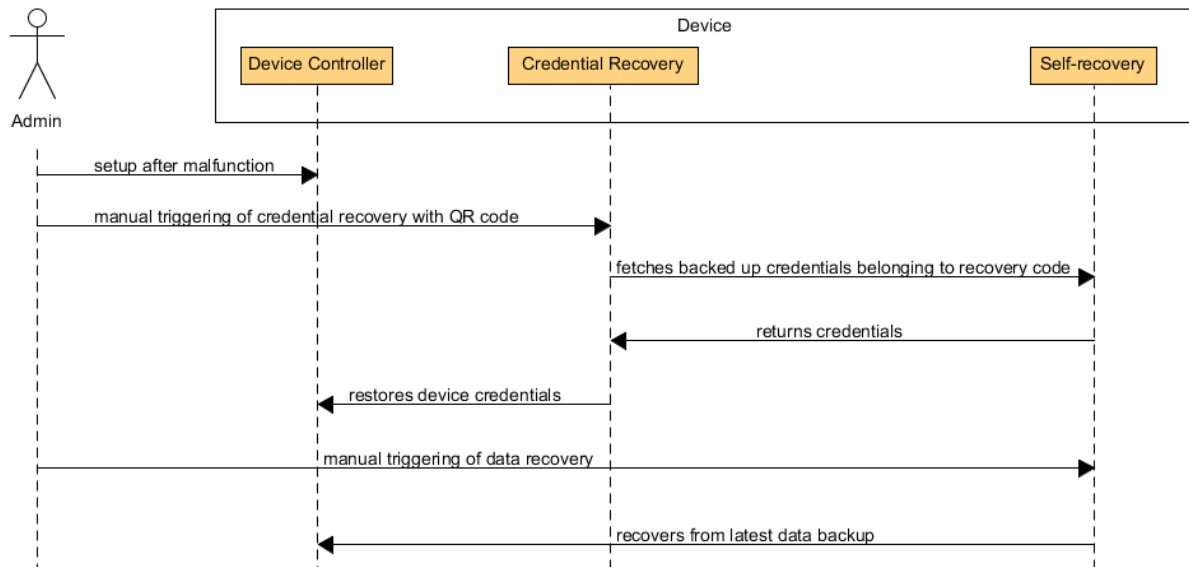


Figure 66 - Simple recovery case

The second diagram represents the case of an automatically triggered recovery process. When a device's reputation score is raised above the threshold of being trustworthy, the on-device components can automatically trigger recovery operations. This sequence also takes into account the requirement of human confirmations of the actions that would happen automatically.

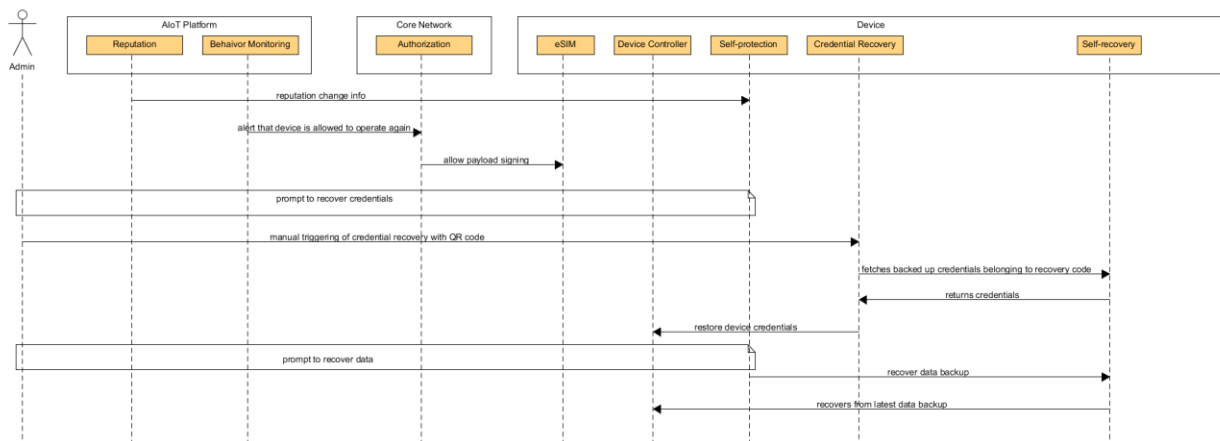


Figure 67 - Auto-triggered recovery

4.1.3.3 Interface description

The REST API of the Self-recovery server allows the client-side recovery script to manage its data. The on-device module is capable of listing the device backups, upload new backups, performing attestation of the uploaded backups and initiate the data recovery process by downloading a backup.

All requests between the client-side and server modules are moderated by the Authentication and Authorization components of the ARCADIAN-IoT platform, verifying the identity and access permission to a set of device backups, supported by the Reputation system, meaning that even

a device with valid credentials may be denied access if its reputation score is too low, indicating the device is compromised.

Self-recovery also communicates with the Reputation system, thus affecting the reputation score of a device. As an example, consistent frequency and size of backups would positively influence the reputation score, while frequent or failed recovery operations would lower it.

The client-side module interfaces with Hardened Encryption (HE) to encrypt backups before they are sent to the server for storage, ensuring data encryption both in transit and at rest. The server is capable of attestation of the uploaded backups, verifying their integrity, but is not capable of decryption: backups can only be unpacked by the device, or an actor with sufficient level of access defined by the HE encryption policy.

In addition to the encryption of backups, payloads are also signed by the on-device eSIM components, adding another layer of security to the communication channel between the IoT device and the server component, residing on the ARCADIAN-IoT platform.

While consistent backup operations can be achieved with a simple job scheduling tool, such as crontab, triggering recovery operations is more complex. A recovery operation may be initiated manually by an administrator, in cases of maintenance or repair of hardware malfunctions, or by an event notification of the Device Self-protection component. The latter case is also split into two options, if the device requires the restoration of identifiers, the credential recovery (see section 4.2) process must be completed before data recovery can commence.

4.1.3.4 API Specification

The final API specification of the Self-recovery server module is described in Swagger OpenAPI format in a YAML file residing on the Gitlab repository⁶⁹ of the project. In addition to providing data recovery capabilities, it also provides helper endpoints for secure storage and recovery of credentials, including a one-time QR-code generator.

4.1.4 Evaluation and results

The final prototype version of the Self-recovery component has been produced. This version is capable of sending data about backup and recovery processes to the Reputation system via RabbitMQ. The backup and recovery process, including encryption and decryption of backed up files, has been tested and validated.

4.2 Credentials recovery

4.2.1 Overview

4.2.1.1 Description

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

Credential Recovery is primarily provided for Self-Sovereign Identity (see sections 2.1 & 3.1),. Additionally, SIM-based Network Identity (see section 2.2) is also considered. Therefore, it is these credentials that are under the scope of being able to be restored. Note that the biometrics credential is not considered for recovery for person credentials as the person is not able to lose them.

Various techniques were evaluated so to automate the process as much as possible while not compromising security.

⁶⁹ https://gitlab.com/arcadian_iot/self-recovery/-/blob/develop/server/api/swagger/swagger.yaml

4.2.1.2 Requirements

A recall of the requirements defined in ARCADIAN-IoT D2.4 [1] with further supplemental information is detailed here.

- **Requirement 7.2.1 – Credentials recovery mechanism**
 - To recover lost, compromised or corrupted credentials for an SSI Agent or Wallet. Analyse also the recovery of network credentials from network operator for authenticating devices/persons in third parties.

4.2.1.3 Objectives and KPIs

The primary objective is to provide the secure recovery of credentials as the first and necessary step to establish the trust before the data recovery mechanism is triggered.

KPI scope	
Support Credential Recovery operations after security/privacy incidents for persons and IoT Devices	
Measurable Indicator	
Credential Recovery mechanisms supported.	
Target value	Achieved value
1	2 ⁷⁰

KPI scope	
Availability of self-recovery and decentralized identity management schemes.	
Measurable Indicator	
Support recovery of Decentralized Identifiers and Verifiable Credentials	
Target value	Achieved value
Recovery supported	Recovery supported

4.2.2 Technology research

4.2.2.1 Background

ATOS does not have any existing assets for credential back-up.

Analysis of the W3C Universal Wallet specification provides for standard interfaces used in a wallet's credential backup and restoration with the use of import and export functions [45]. It also states that if "Encrypted Data Vault" is used then it is not needed to support those functions. However, local encryption at the client side could be a more secure option so that no unencrypted credentials are shared outside of the wallet whatsoever.

In the following sub-sections, we outline the approach followed in ARCADIAN-IoT to recover access credentials for gaining access to the Self-Recovery component to manage the backups.

Credential recovery considers the scenario where a mobile or IoT device's data (including credential SSI Wallet or from an IoT device) was somehow corrupted or lost and the user or

⁷⁰ Different recovery approaches are taken for the wallet and IoT Devices, so 2 mechanisms are supported.

device is attempting a recovery of its credentials and later its data. It was considered the recovery of the credentials in an automatic way as much as possible before the data back-up could be securely accessed.

The credentials that are backed up make use of cipher techniques, at source, to encrypt the credentials data.

Support for the recovery of credentials for mobile and IoT devices are analysed in the following section.

4.2.2.2 Credentials Recovery in ARCADIAN-IoT

4.2.2.2.1 Person Credentials Recovery

The different person identity credentials supported in ARCADIAN-IoT and their possible recovery mechanisms are analysed below.

Credentials Recovery for a SSI Wallet on a mobile device

For recovery of a user's credentials, the following mechanisms for a user's SSI Wallet can be considered:

- a secure back-up server can be setup a quorum of trusted entities associated emails to be issued with different portions of a recovery key to be used to request a restoration of the SSI Wallet including its Verifiable Credentials.
- a secure back-up server can issue a QR Code containing a recovery key based on its authenticated identity to be used later to request the restoration of the SSI Wallet.
- The SSI Wallet can request the user a secret and email address to be used to store a backup and later identify a backup for restoration of the SSI Wallet.

The SSI Wallet backup is encrypted, and the restoration process described above provides access to the key to decrypt it and restore the wallet. Secure symmetric encryption such as AES 256 is one way where the wallet could be encrypted and exported to a recovery server where it would be stored with a specific recovery key.

Credentials Recovery for SIM-based identity on a mobile device

The root cause that could trigger the SIM credentials recovery in ARCADIAN-IoT is a mobile device being stolen or lost.

If the device itself was later recovered (its owner has it again without being irreparably shattered), the SIM credentials continue there, in the secure element. The threat surface related with the access to the SIM credentials and compromising them is very narrow and has very low probability. SIMs are well-accepted as secure to store and manage subscriber identity credentials for decades. Furthermore, in fact, the use of eSIM instead of the previous SIM factor further narrows the threat surface, because the eUICC (the hardware that has the information – SIM profiles) is soldered to the device board, being much harder for, e.g., removing it for cloning when compared to the previous plastic cards.

Therefore, considering the SIM/eSIM secure element (UICC/eUICC) as being secure, for the ARCADIAN-IoT user network-based credentials recovery it is only considered the case that the device is not recovered by its owner. In this case, a new device will have a new SIM profile (new subscriber credentials) and the credentials recovery will consist of associating these new credentials with the previous ARCADIAN-IoT ID of the impacted person. This process will be similar to the onboarding process in what concerns the SIM credentials association with an ARCADIAN-IoT ID, with the instruction to update the onboarding with the new SIM. To ensure trust in this process, new SIM/network credentials can only be associated with an existing ARCADIAN-IoT ID after the recovery of the SSI previously described.

4.2.2.2.2 IoT Device Credential Recovery

The different IoT Device identity credentials supported in ARCADIAN-IoT and their possible recovery mechanisms are analysed below.

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

Once an IoT Device is registered with an ARCADIAN-IoT identity, the controller (Service Provider application) of the IoT Device could request a recovery key from Self-recovery and use this in a request for performing Credential Recovery. The IoT Device would then contact Self-Recovery with the recovery key to recover its previous credentials.

In the case of IoT devices that lost their DID or where the private key was compromised, the DID DOC would have to be first updated / recovered (as described in section 2.1) with the IoT device updating its associated private key. However, as the IoT Device is not assumed to be issued with so many different credentials as a person is, a simpler approach is recommended to restore the credentials with a the business app logic requesting to update the DID by rotating its keys and re-issuing its device Verifiable Credential being unique to the device fingerprint.

Credentials Recovery for eSIM on a IoT device

The credentials recovery for the cellular network subscriber should be the same for both IoT devices or personal devices, and only applies when devices' hardware is not recovered. Upon a recovery process, a new SIM profile is provisioned to the substitute device and the new credentials are associated with the ARCADIAN-IoT entity being recovered.

This scenario is out of scope as no new device will actually be re-onboarded but will instead have its own onboarding, as each device has a different fingerprint.

4.2.2.3 Research findings and achievements

The initial research findings proposed 3 ways to provide the import and export of encrypted back-ups of the wallet credentials. For simplicity the solution chosen is the one based on a user-provided secret where the hash of the secret is used to identify the backup file in the back-up server.

The recovery of IoT Devices credentials selected the more automated approach to instruct the IoT Device to perform a key rotation and be re-issued with its device Verifiable Credential, which is based on its hardware fingerprint and as such will not change.

4.2.2.4 Produced resources

Person credential recovery implementation is built on the uSelf Wallet and supported with a new dedicated back-up server for the Ledger uSelf solution.

IoT Device credential recovery implementation is built on the SSI GO Agent, developed in Verifiable Credentials component in 3.1.

4.2.3 Design specification

The design of person credential recovery is described in this deliverable.

4.2.3.1 Sub-use cases

4.2.3.1.1 Recovery of a person's SSI Credentials

The use case figure below shows the recovery of person credentials from a user's SSI wallet.

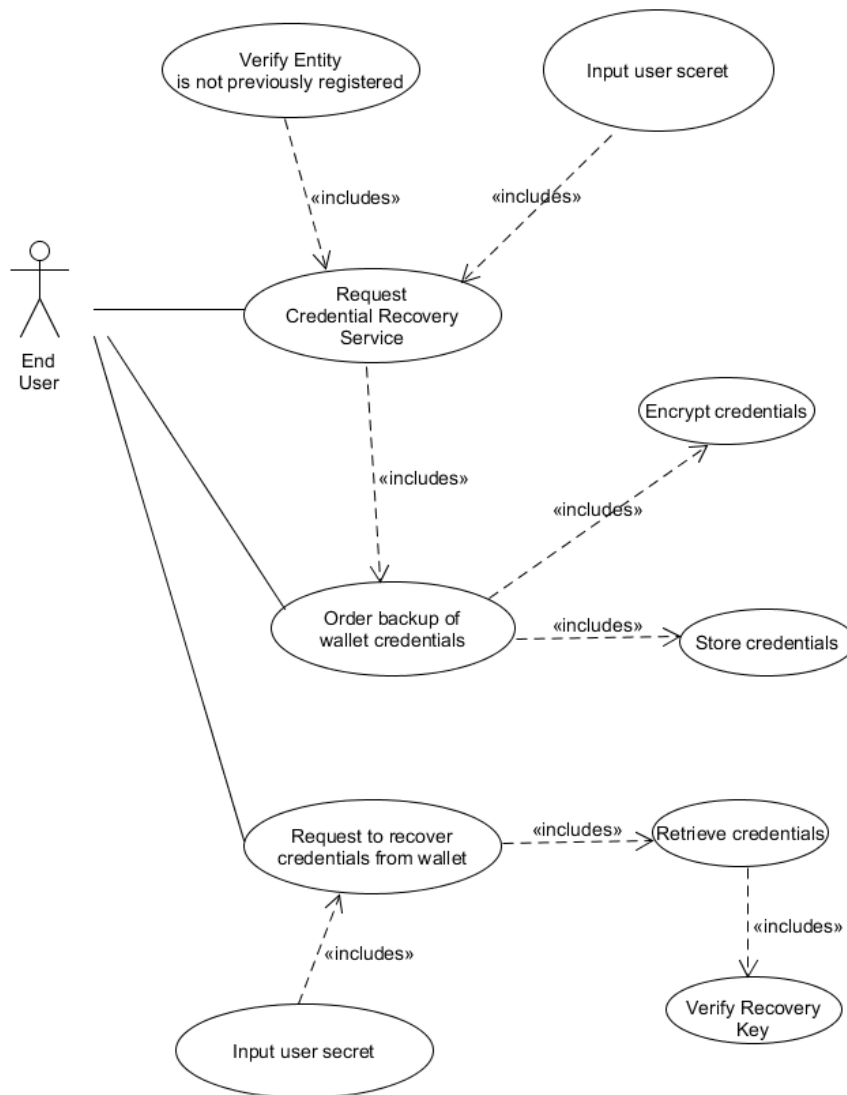


Figure 68 - Recovery of SSI Wallet Credentials Use Case

4.2.3.1.2 Recovery of an IoT Device's SSI Credentials

The figure below captures the use case for supporting the recovery of an IoT Device's credential by rotating the DIDs keys and being re-issued with its Device VC. This is expected to happen after a factory reset of the device.

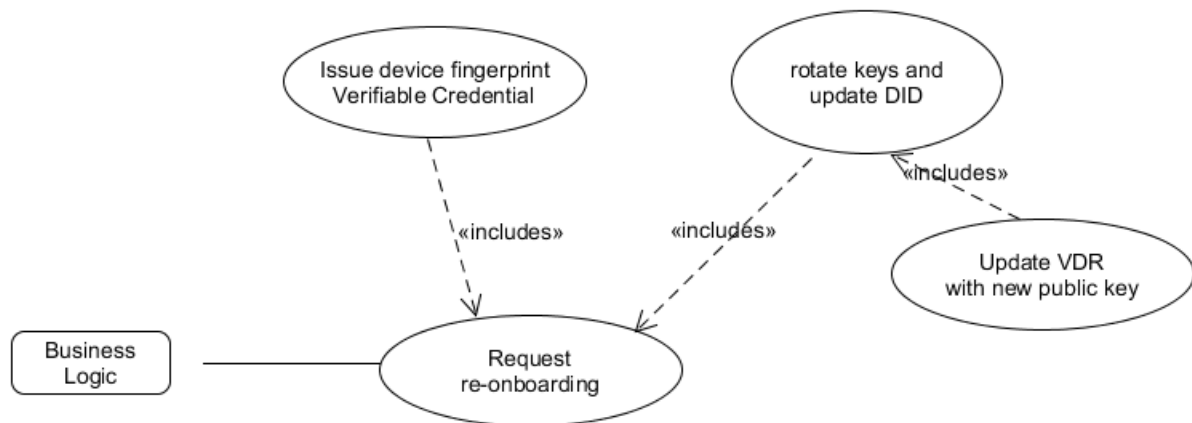


Figure 69 Recovery of IoT Device Credentials Use Case

4.2.3.2 Logical architecture view

4.2.3.2.1 SSI Wallet Credential Recovery

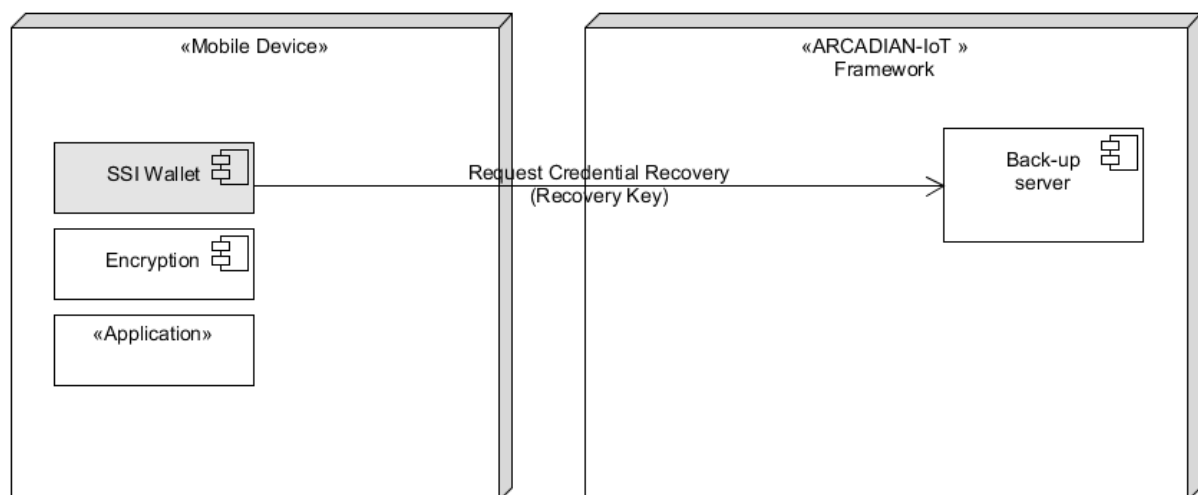


Figure 70 - SSI Credential Recovery Logical Architecture

4.2.3.2.2 IoT Device Credential Recovery

The logical architecture is not changed from the Ledger uSelf solution and in particular the SSI GO Agent provided in section 3.1.3.2.

4.2.3.3 Sequence diagrams

4.2.3.3.1 SSI Wallet Credential Recovery

No sequence diagram was seen to be needed due to its simple interwork with the uSelf's own back-up server.

4.2.3.3.2 SIM Credentials Recovery for ARCADIAN-IoT person

No sequence diagram is provided as this follows the SP update of a registered person flow as provided in section 3.1.3.3.14.

4.2.3.3 IoT Device Credential Recovery

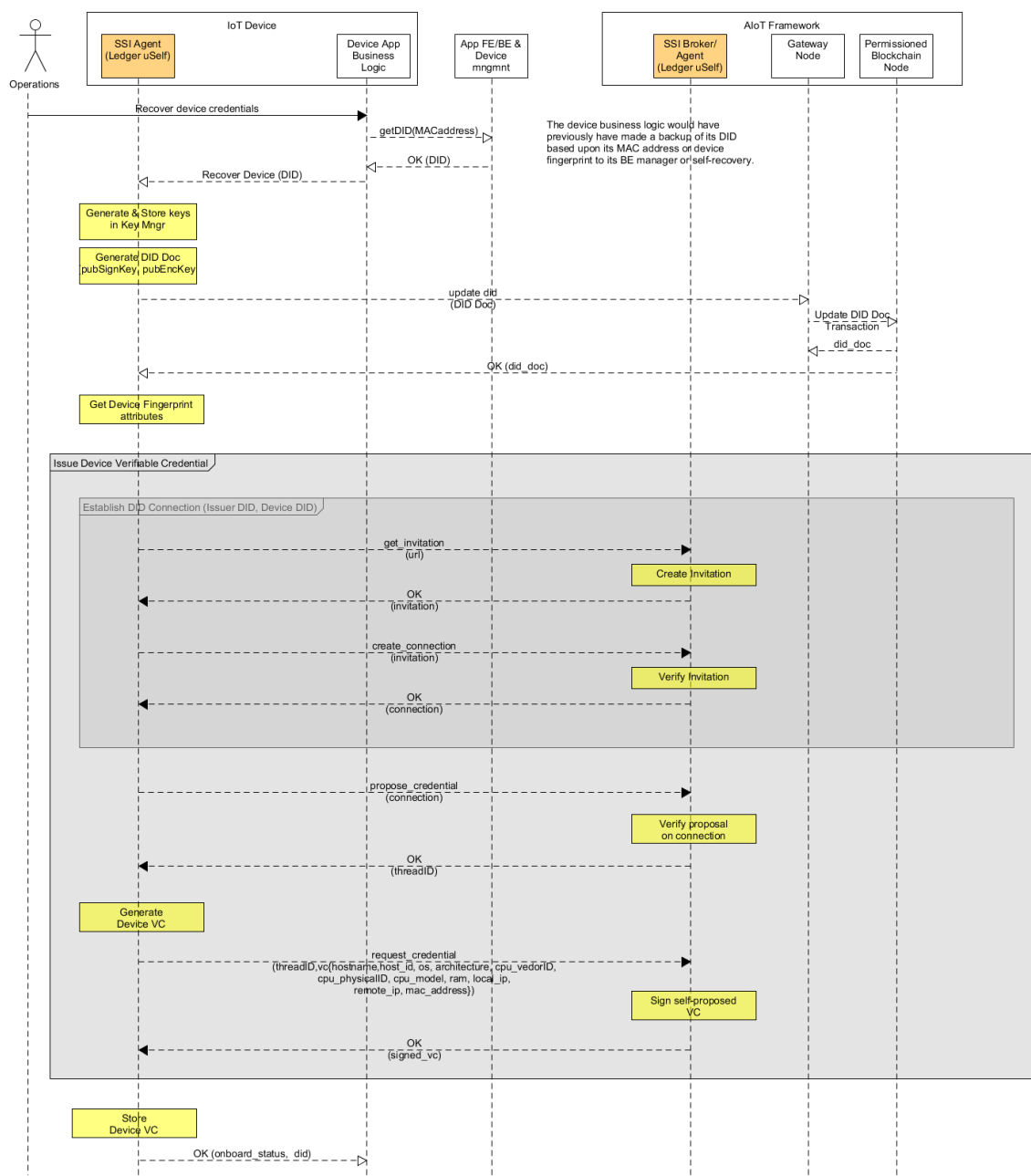


Figure 71 IoT Device Credential Recovery

4.2.3.4 Interface description

No interface description is given for the back-up server as it is an in-house development and not needed to be shared with any 3rd party.

4.2.3.5 Technical solution

4.2.3.5.1 API specification

The store / retrieve credential back-up calls are provided by a dedicated back-up server so there is no need to publish them to any other component, and therefore not made available.

4.2.3.5.2 *Frontend design*

The SSI Wallet will provide a user interface for enabling the user to:

1. Turn on the Credential Recovery service and provide email address
2. Request the user secret used with email for encrypting the credential back-up and generating a hash used as an index in the back-up server to identify the back-up file.
2. Order a credential back-up with the user provided secret key and email.
3. Order a credential recovery with the user secret and email to request credential file and decrypt it.

4.2.3.5.3 *Ledger uSelf mobile SSI wallet*

The Ledger uSelf mobile SSI wallet supports:

- the symmetric encryption of the SSI credentials using the android encryption libraries.
- the storing of encrypted credential backups to a dedicated back-up server.
- the secure retrieval of the encrypted credential backup from the back-up server.

4.2.4 Evaluation and results

Test validation was successfully carried out internally by ATOS for SSI wallet and IoT Device recovery, and component integration for SIM recovery and pilot validation is carried out in WP5.

4.2.5 Future work

To carry out a successful demonstration in ARCADIAN-IoT of Credential-Recovery, and learn from user and stakeholder feedback, and to exploit it in the Ledger uSelf toolset.

5 CONCLUSIONS

This report has aimed at presenting the outcomes from the research associated to ARCADIAN-IoT's Vertical Planes, i.e. Identity, Trust and Recovery planes. It covered both the background, research and specifications, implementation and any performed evaluation for each of the components, namely:

- Decentralized Identifiers, eSIM for hardware-based identity and authentication, Biometrics and Authentication, which establish the Identity plane
- Verifiable Credentials, Network-based Authorization, Reputation System and Remote Attestation, which form the Trust plane
- Self-recovery and Credentials Recovery, forming the Recovery plane

The Trust and Identity planes have a paramount role in enabling ARCADIAN-IoT's Chain of Trust for the different IoT device categories, while the Recovery plane provides essential abilities needed for ensured data integrity in case of incidents.

This final iterative version of the deliverable (based on deliverable D4.2) reflects the Vertical plan features enabled in the final P2 prototype, where the partners concluded the technological research of each component and updated their design for added features and improvements.

Most of the KPIs associated to the innovation targets from each component were reached, while a few still depend on the actual P2 validation in the domain pilots, which is going to be documented in D5.5 at M36 of the project.

APPENDIXES

Appendix A – Analysis of events in Domain A for Reputation System

ID	Event	Use Case as per D2.2 [2] Description	Reputation Ratings Logic Initially: - Persons reputation NULL - Services reputation NULL - Devices reputation NULL	PERSON Reputation Ratings Values (min 0.0, max 1.0)	Service Reputation Ratings Values (min 0.0, max 1.0)	Device Reputation Ratings Values (min 0.0, max 1.0)	Device 2 (Drone/other) Reputation
E1	Person Registration	A1– Person registers in the DGA service and install mobile APP,	User register in service	+		n/a	

Appendix B – Analysis of events in Domain B for Reputation System

ID	Event	Use Case as per D2.2 [2] Description	Reputation Ratings Logic Initially: - Persons reputation NULL - Services reputation NULL - Devices reputation NULL	Device Reputation Ratings Values (min 0.0, max 1.0)	Service/Middleware Reputation Ratings Values (min 0.0, max 1.0)	Person Reputation (Grid Manager) Ratings Values (min 0.0, max 1.0)
A1	New user (human being) registration	GRID	Persons reputation NULL; (+) or maintain at existing level	N/A	N/A	0,1

Appendix C – Analysis of events in Domain C for Reputation System

ID	Event	Use Case as per D2.2 [2] Description	Reputation Ratings Logic Initially: - Persons reputation NULL - Services reputation NULL - Devices reputation NULL	Medical Person Reputation Ratings Values (min 0.0, max 1.0)	Patient Person Reputation Ratings Values (min 0.0, max 1.0)	Service Reputation Ratings Values (min 0.0, max 1.0)	Device Reputation Ratings Values (min 0.0, max 1.0)	MIoT Hospital platform Reputation Rating values (min: 0.0, max 1.0)
E 1.	MIoT kit delivery - Patient registration and authentication	C1	User register in service		+	+		
			Patient Authenticates		+	+	+	
			Patient Fails Authentication		-		-	

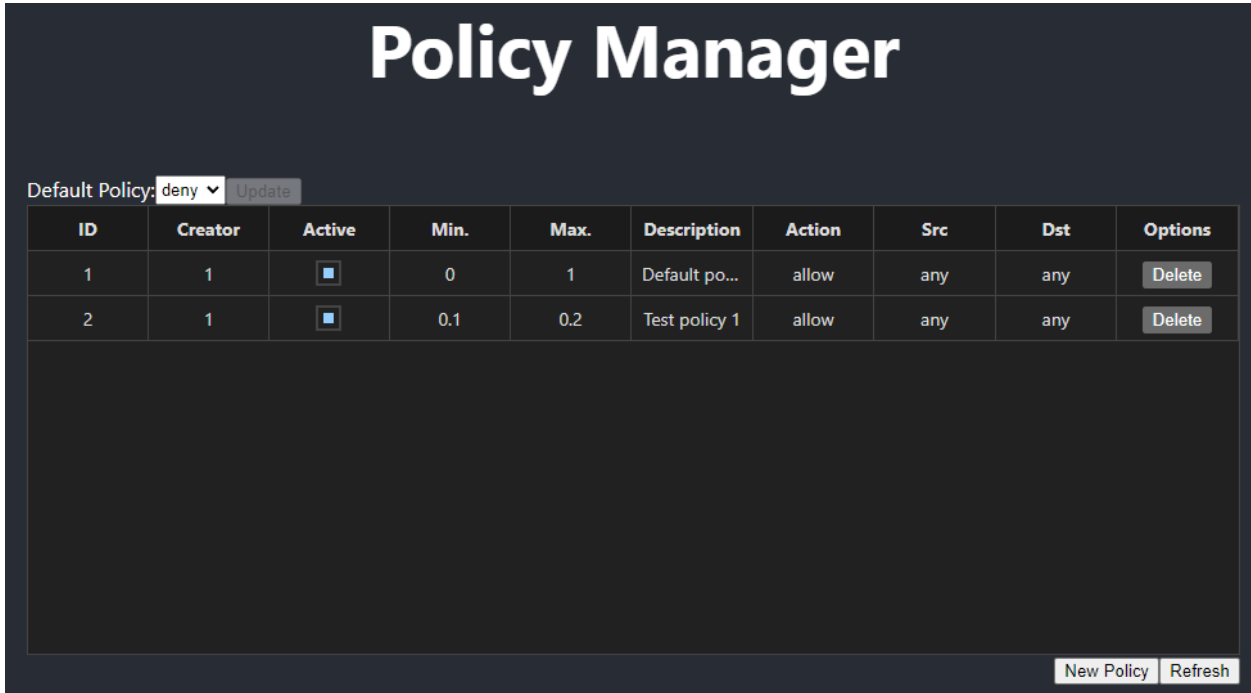
Appendix D – Policy Manager User Manual

Policy Management Dashboard

Default policy

Default policy is a policy which specifies the default behaviour of the system when no other policies are applied. **There can only be one default policy.**

In order to update the default policy simply select a different value on the drop-down menu and click submit.




ID	Creator	Active	Min.	Max.	Description	Action	Src	Dst	Options
1	1	<input checked="" type="checkbox"/>	0	1	Default po...	allow	any	any	Delete
2	1	<input type="checkbox"/>	0.1	0.2	Test policy 1	allow	any	any	Delete

Figure 72 - Screen with default and other policies

Updating policies

In order to update a policy follow the steps:

1. Select a field on the table
2. Change the value
3. Click off the field or press “Enter”



Policy Manager

Active*: ☐

Min. Reputation Score*:

Minimum Score

Max. Reputation Score*:

Maximum Score

Description:

Description

Policy Action*:

allow

Src ID:

Src ID

Dst ID:

Dst ID

Figure 73 – Configurable policies fields

Deleting policies

In order to delete a policy follow the steps:

1. Scroll to the desired policy
2. Click on the “Delete” button on the last field of the row

Adding new policies

In order to add a new policy to the database follow the steps:

1. Click on the “New Policy” button
2. Fill the fields
3. Click on the “Add Policy” button

Policy API

Policy fields

Backend fields

- Id
 - **Policy ID**
- createdBy
 - **Policy creator’s ID**
- timeUpdated
 - **Timestamp of the last changes to the policy**

Policy definition fields

- Active (**mandatory**)
 - **Boolean representing whether the policy is active or not**
- reputationRange (**mandatory**)
 - **Reputation range at which the policy is active**
 - **Range [min, max]**
- Description (**optional**)

- **Optional text description of the policy**
- **Action (mandatory)**
 - **Policy effect (allow or deny)**
- **SrcIDs (optional)**
 - **Array with IDs of the domain targeted by the policy**
 - **Can include: AIoT identifiers,**
- **DstIDs (optional according to policy)**
 - **Array with IDs of the destination domain**
 - **Can include: AIoT identifiers, IP addresses, FQDN**

Endpoints

GET /policies

POST /policy

PATCH /policy/{id}

DELETE /policy/{id}

GET /default_policy

PUT /default_policy

Appendix E – DID METHODS

Table 13 DID Method Table [8]

DID Method	DLT / Network	Name
did:3	Ceramic Network	3ID DID Method
did:abt	ABT Network	ABT DID Method
did:aergo	Aergo	Aergo DID Method
did:ala	Alastria	Alastria DID Method
did:amo	AMO blockchain mainnet	AMO DID Method
did:bba	Ardor	BBA DID Method
did:bid	bif	BIF DID Method
did:bnb	Binance Smart Chain	Binance DID Method
did:bryk	bryk	bryk DID Method
did:btc	Bitcoin	BTCDID Method
did:ccp	Quorum	Cloud DID Method
did:celo	Celo	Celo DID Method
did:com	commercio.network	Commercio.network DID Method
did:corda	Corda	Corda DID method
did:did	Decentralized Identifiers	DID Identity DID Method
did:dns	Domain Name System (DNS)	DNS DID Method
did:dock	Dock	Dock DID Method
did:dom	Ethereum	
did:dual	Ethereum	Dual DID Method
did:echo	Echo	Echo DID Method
did:elastos	Elastos ID Sidechain	Elastos DID Method
did:elem	Element DID	ELEM DID Method
did:emtrust	Hyperledger Fabric	Emtrust DID Method
did:ens	Ethereum	ENS DID Method
did:eosio	EOSIO	EOSIO DID Method
did:erc725	Ethereum	erc725 DID Method
did:etho	Ethereum	ETHO DID Method
did:ethr	Ethereum	ETHR DID Method
did:evan	evan.network	evan.network DID Method
did:example	DID Specification	DID Specification
did:factom	Factom	Factom DID Method
did:future	Netease Chain	Future DID Method
did:gatc	Ethereum, Hyperledger Fabric, Hyperledger Besu, Alastria	Gataca DID Method
did:grg	GrgChain	GrgChain DID Method
did:hedera	Hedera Hashgraph	Hedera Hashgraph DID Method
did:holo	Holochain	Holochain DID Method
did:hpass	Hyperledger Fabric	hpass DID Method
did:icon	ICON	ICON DID Method
did:infra	InfraBlockchain	Infra DID Method
did:io	IoTeX	IoTeX DID Method
did:ion	Bitcoin	ION DID Method
did:iota	IOTA	IOTA DID Method
did:ipid	IPFS	IPID DID method
did:is	Blockcore	Blockcore DID Method

did:iw	InfoWallet	InfoWallet DID Method
did:jline:	JLINC Protocol	JLINC Protocol DID Method
did:jnctn	Jnctn Network	JNCTN DID Method
did:jolo	Ethereum	Jolocom DID Method
did:keri	Ledger agnostic	KERI DID Method
did:key	Ledger independent DID method based on public/private key pairs	DID key method
did:kilt	KILT Blockchain	KILT DID Method
did:klay	Klaytn	Klaytn DID Method
did:kr	Korea Mobile Identity System	Korea Mobile Identity System DID Method
did:lac	LACChain Network	LAC DID Method
did:life	RChain	lifeID DID Method
did:lit:	LEDGIS	LIT DID Method
did:meme	Ledger agnostic	Meme DID Method
did:meta	Metadium	Metadium DID Method
did:moac	MOAC	MOAC DID Method
did:monid	Ethereum	MONiD DID Method
did:morpheus	Hydra	Morpheus DID Method
did:mydata	iGrant.io	Data Agreement DID Method
did:near	NEAR	NEAR DID Method
did:nft	Ceramic Network	NFT DID Method
did:ockam	Ockam	Ockam DID Method
did:omn	OmniOne	OmniOne DID Method
did:onion	Ledger agnostic	Onion DID Method
did:ont	Ontology	Ontology DID Method
did:op	Ocean Protocol	Ocean Protocol DID Method
did:orb	Ledger agnostic	Orb DID Method
did:panacea	Panacea	Panacea DID Method
did:peer	peer	peer DID Method
did:pistis	Ethereum	Pistis DID Method
did:pkh	Ledger-independent generative DID method based on CAIP-10 keypair expressions	did:pkh method
did:pml	PML Chain	PML DID Method
did:polygon	Polygon (Previously MATIC)	Polygon DID Method
did:ptn	PalletOne	PalletOne DID Method
did:safe	Gnosis Safe	SAFE DID Method
did:san	SAN Cloudchain	SAN DID Method
did:schema	Multiple storage networks, currently public IPFS and evan.network IPFS	Schema Registry DID Method
did:selfkey	Ethereum	SelfKey DID Method
did:sideos	Ledger agnostic	sideos DID Method
did:signor	Ethereum, Hedera Hashgraph, Quorum, Hyperledger Besu	Signor DID Method
did:sirius	ProximaX Sirius Chain	ProximaX SiriusID DID Method
did:sol	Solana	SOL DID Method
did:sov	Sovrin	Sovrin DID Method
did:ssb	Secure Scuttlebutt	SSB DID Method

did:ssw	Initial Network	SSW DID Method
did:stack	Bitcoin	Blockstack DID Method
biodid:tangle	IOTA Tangle	TangleID DID Method
did:tls	Ethereum	TLS DID Method
did:trust	TrustChain	Trust DID Method
did:trustbloc	Hyperledger Fabric	TrustBloc DID Method
did:trx	TRON	TRON DID Method
did:ttm	TMChain	TM DID Method
did:twit	Twit	Twit DID Method
did:tyron	Zilliqa	tyronZIL DID-Method
did:tys	DID Specification	TYS DID Method
did:tz:	Tezos	Tezos DID Method
did:unik	uns.network	UNIK DID Method
did:unisot	Bitcoin SV	UNISOT DID Method
did:uns	uns.network	UNS DID Method
did:uport	Ethereum	
did:v1	Veres One	Veres One DID Method
did:vaa	bif	VAA Method
did:vaultie	Ethereum	Vaultie DID Method
did:vid	VP	VP DID Method
did:vivid	NEO2, NEO3, Zilliqa	Vivid DID Method
did:vvo	Vivvo	Vivvo DID Method
did:web	Web	Web DID Method
did:wlk	Weelink Network	Weelink DID Method
did:work	Hyperledger Fabric	Workday DID Method

REFERENCES

- [1] ARCADIAN-IoT, “D2.4 ARCADIAN-IoT framework requirements”, 2021
- [2] ARCADIAN-IoT, “D2.2 - Use case specification”, 2021
- [3] DIF, “Decentralized Identifiers (DIDs) v1.0,» 03 Aug 2021.”, <https://www.w3.org/TR/did-core/>, 2022
- [4] DIF, “Sidetree v1.0.0”, <https://identity.foundation/sidetree/spec/>, 2022
- [5] ARCADIAN-IoT, “D3.1 - Horizontal Planes - first version”, 2022
- [6] W3C, “BBS+ Signatures 2020”, <https://w3c-ccg.github.io/ldp-bbs2020/>, 2022
- [7] W3C, “Verifiable Credentials Data Model v1.1”, <https://www.w3.org/TR/vc-data-model/>, 2022
- [8] W3C, “DID Registries”, <https://www.w3.org/TR/did-spec-registries/>, 2022
- [9] <https://github.com/decentralized-identity/element/blob/master/docs/did-method-spec/spec.md>, 2022
- [10] <https://github.com/w3c-ccg/did-method-web>, 2022
- [11] <https://github.com/decentralized-identity/ion-did-method>, 2022
- [12] <https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/DID+Registry+API>, 2022
- [13] “IOTA DID Method specification”, https://wiki.iota.org/identity.rs/specs/did/iota_did_method_spec, 2022
- [14] <https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/DID+Authentication+Library>, 2022
- [15] “Sidetree Core Protocol and DID Method Drivers“, <https://github.com/transmute-industries>, 2022
- [16] Transmute, “Sidetree Protocol Specification”, <https://github.com/transmute-industries/sidetree-core/blob/master/docs/protocol.md>
- [17] <https://w3c-ccg.github.io/did-method-key/>, 2022
- [18] <https://identity.foundation/peer-did-method-spec/index.html>, 2022
- [19] <https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Early+Adopters+Programme>, 2022
- [20] <https://joinup.ec.europa.eu/collection/ssi-eidas-bridge/about>, 2022
- [21] Evernym, <https://www.evernym.com/blog/bbs-verifiable-credentials/>, 2022
- [22] DIF, <https://identity.foundation/didcomm-messaging/spec/>, 2022
- [23] <https://github.com/uport-project/veramo>, 2022
- [24] <https://veres.one/>, 2022
- [25] <https://www.hyperledger.org/use/hyperledger-indy>, 2022
- [26] <https://github.com/hyperledger/aries>, 2022

- [27] <https://jolocom.io/>, 2022
- [28] <https://github.com/matttrglobal>, 2022
- [29] <https://github.com/spruceid/ssi>, 2022
- [30] <https://github.com/iotaledger/identity.rs/>, 2022
- [31] <https://github.com/alastria/alastria-identity>, 2022
- [32] <https://github.com/decentralized-identity/interoperability/blob/master/agenda2021.md>, 2022
- [33] <https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/EBSI+Verifiable+Credentials+Playbook>, 2022
- [34] https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0.html, 2022
- [35] https://openid.net/specs/openid-connect-self-issued-v2-1_0.html, 2022
- [36] <https://datatracker.ietf.org/doc/html/draft-looker-jwm-01>, 2022
- [37] <https://w3c.github.io/vc-test-suite/implementations/>, 2022
- [38] <https://github.com/transmute-industries/sidetree.js>, 2022
- [39] <https://www.ietf.org/archive/id/draft-cavage-http-signatures-06.txt>, 2022
- [40] W3C “Decentralized Identifiers (DIDs)” <https://w3c-ccg.github.io/did-spec/>
- [41] W3C Peer DID Method specification, <https://identity.foundation/peer-did-method-spec/index.html>, 2022
- [42] <https://github.com/hyperledger/aries-framework-go>, 2022
- [43] <https://github.com/hyperledger/aries-framework-go/graphs/contributors>, 2022
- [44] https://ec.europa.eu/info/strategy/priorities-2019-2024/europe-fit-digital-age/european-digital-identity_en, 2022
- [45] <https://w3c-ccg.github.io/universal-wallet-interop-spec/#import>, 2022
- [46] ARCADIAN-IoT, “D4.1 - Vertical Planes”, 2022
- [47] ARCADIAN-IoT, “D3.3 - Horizontal Planes - first version”, 2024
- [48] DID Challenge / Response Authentication <https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/%5Barchived%5DTechnical+Specification+%2810%29+-+Access+to+Relying+party+using+DID-Auth>, 2023
- [49] Hyperledger Aries RFC 0646 BBS+ Credentials, <https://github.com/hyperledger/aries-rfcs/blob/main/features/0646-bbs-credentials/README.md>, 2023
- [50] BBS+ Signature Proof Suite 2020, <https://w3c-ccg.github.io/ldp-bbs2020/>, 2023
- [51] EBSI Natural Person VerifiableID, <https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Verifiable+ID+-+Natural+Person>, 2023

- [52] Data Spaces Convergence paper, https://data-spaces-business-alliance.eu/wp-content/uploads/dlm_uploads/Data-Spaces-Business-Alliance-Technical-Convergence-V2.pdf . 2023
- [53] <https://digital-strategy.ec.europa.eu/en/news/eu-digital-identity-4-projects-launched-test-eudi-wallet>, 2023
- [54] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedures (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <<https://www.rfc-editor.org/rfc/rfc9334>>.
- [55] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/rfc/rfc8949>>.
- [56] 3GPP TS 43.020 version 15.0.0 Release,
https://www.etsi.org/deliver/etsi_ts/143000_143099/143020/15.00.00_60/ts_143020v150000p.pdf
- [57] Global Information Assurance Certification, The GSM Standard (An overview of its security), [https://www.giac.org/paper/gsec/1499/gsm-standard-an-overview-security/102787#:~:text=GSM%20makes%20use%20of%20a,a%20cipherring%20key%20\(KC\)](https://www.giac.org/paper/gsec/1499/gsm-standard-an-overview-security/102787#:~:text=GSM%20makes%20use%20of%20a,a%20cipherring%20key%20(KC))
- [58] T. Silva, J. Casal and R. Chaves, "Lightweight network-based IoT device authentication in Cloud services," in 31st IEEE International Conference on Network Protocols, 2023.
- [59] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of internet of things (iot) authentication schemes," Sensors, vol. 19, no. 5, p. 1141, 2019.
- [60] K. Nimmy, S. Sankaran, K. Achuthan, and P. Calyam, "Lightweight and privacy-preserving remote user authentication for smart homes," IEEE Access, vol. 10, pp. 176–190, 2021.
- [61] M. Yoda, S. Sakuraba, Y. Sei, Y. Tahara, and A. Ohsuga, "Detecting hardcoded login information from user input," in IEEE International Conference on Consumer Electronics (ICCE). IEEE, 2022, pp. 1–2.
- [62] Y. Atwady and M. Hammoudeh, "A survey on authentication techniques for the internet of things," in Proceedings of the International Conference on Future Networks and Distributed Systems, 2017.
- [63] J. Mamvong, "Efficient security algorithm for provisioning constrained internet of things (iot) devices," 2023.
- [64] M. A. Jan, F. Khan, M. Alam, and M. Usman, "A payload-based mutual authentication scheme for internet of things," Future Generation Computer Systems, vol. 92, pp. 1028–1039, 2019.
- [65] E. Lara, L. Aguilar, M. A. Sanchez, and J. A. García, "Lightweight authentication protocol for m2m communications of resource-constrained devices in industrial internet of things," Sensors, vol. 20, no. 2, p. 501, 2020.
- [66] M. T. Hammi, E. Livolant, P. Bellot, A. Serhrouchni, and P. Minet, "A lightweight mutual authentication protocol for the iot," in International Conference on Mobile and Wireless Technology. Springer, 2017.
- [67] N. Ye, Y. Zhu, R.-c. Wang, R. Malekian, and Q.-m. Lin, "An efficient authentication and access control scheme for perception layer of internet of things," 2014.
- [68] L. Seitz, G. Selander, E. Wahlstroem, S. Erdtman, and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)," Internet Engineering Task Force, Internet-Draft draft-ietf-ace-oauth-authz-42, Jun. 2021, work in Progress. [Online]. Available:<https://datatracker.ietf.org/doc/html/draft-ietf-ace-oauth-authz-42>

- [69] IETF, “The oauth 2.0 authorization framework,” Internet Standard, October 2012, <https://datatracker.ietf.org/doc/html/rfc6749>.
- [70] Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (coap),” Internet Requests for Comments, RFC Editor, RFC 7252, June 2014. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc7252.txt>
- [71] M. Jones, E. Wahlstroem, S. Erdtman, and H. Tschofenig, “Cbor web token (cwt),” Internet Requests for Comments, RFC Editor, RFC 8392, May 2018. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc8392.txt>
- [72] P. Hoffman and C. Bormann, “Concise binary object representation (cbor),” Internet Requests for Comments, RFC Editor, RFC 8949, December 2020. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc8949.txt>
- [73] H. Flores, X. Su, V. Kostakos, A. Y. Ding, P. Nurmi, S. Tarkoma, P. Hui, and Y. Li, “Large-scale offloading in the internet of things,” in 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). IEEE, 2017.
- [74] S. U. Malik, H. Akram, S. S. Gill, H. Pervaiz, and H. Malik, “Effort: Energy efficient framework for offload communication in mobile cloud computing,” *Software: Practice and Experience*, vol. 51, no. 9, pp. 1896–1909, 2021.
- [75] GSMA, “Security features of lte-m and nb-iot networks,” GSMA, <https://www.gsma.com/iot/wp-content/uploads/2019/09/Security-Features-of-LTE-M-and-NB-IoT-Networks.pdf>, Mobile IoT Security Report, September 2019.
- [76] A. Y. Korkusuz, “Security in the gsm network,” Term project, vol. 2012, pp. 1–34, 2012.
- [77] C. H. B. Morgado and M. Moorfield, “Authentication of devices to third party services,” Patent Cooperation Treaty Patent PCT/GB2021/051 093, Nov. 11, 2021.