



Grant Agreement N°: 101020259

Topic: SU-DS02-2020



ARCADIAN-IoT

Autonomous Trust, Security and Privacy
Management Framework for IoT

D4.2: ARCADIAN-IoT Vertical Planes – 2nd version

Revision: v1.0

Work package	4
Task	Tasks 4.1, 4.2, 4.3, 4.4, and 4.5
Due date	31/12/2022
Submission date	30/12/2022
Deliverable lead	IPN
Version	1.0
Partner(s) / Author(s)	ATOS: Ross Little IPN: Sérgio Figueiredo, Rúben Leal TRU: João Casal, Carlos Morgado, Tomás Silva, José Rosa, Ivo Vilas Boas UWS: Jose M. Alcaraz Calero, Qi Wang, Ignacio Martinez-Alpiste, Gelayol Golcarenarenji, Julio Diez Tomillo, Mohamed Khadmaoui-Bichouna, Mohammad Alselek XLAB: Jan Antic UC: Bruno Sousa, Luis Paquete, Miguel Arieiro, João Nunes

Abstract

This public technical report constitutes deliverable D4.2 of ARCADIAN-IoT, a Horizon2020 project with the **grant agreement number 101020259**, under the topic **SU-DS02-2020**. D4.2 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)
V0.1	06/09/2022	Preliminary Template	ATOS
V0.2	30/11/2022	Multiple partners contributions	ATOS, IPN, UC, TRU, XLAB, UWS, XLAB, UC
V0.3	05/12/2022	Deliverable quality revision	IPN
V0.4	21/12/2022	Internal review	ATOS
V0.5	27/12/2022	Document cleaning and formatting	IPN
V1.0	30/12/2022	Conclusions review and final editing of the document	IPN

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 D4.2 is the second 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.

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	10
ABBREVIATIONS	11
1 INTRODUCTION	13
1.1 ARCADIAN-IoT and its Vertical Planes.....	13
1.2 Objectives	14
2 IDENTITY PLANE	16
2.1 Decentralized Identifiers (ATOS)	16
2.1.1 Overview.....	16
2.1.2 Technology research	18
2.1.3 Design specification	25
2.1.4 Evaluation and results.....	33
2.1.5 Future work.....	34
2.2 eSIM – Hardware-based identification and authentication (TRU)	35
2.2.1 Overview.....	35
2.2.2 Technology research	37
2.2.3 Design specification	39
2.2.4 Evaluation and results.....	41
2.2.5 Future work.....	41
2.3 Biometrics (UWS)	41
2.3.1 Overview.....	41
2.3.2 Technology research	44
2.3.3 Design specification	47
2.3.4 Evaluation and results.....	52
2.3.5 Future work.....	52
2.4 Authentication (TRU)	53
2.4.1 Overview.....	53
2.4.2 Technology research	54
2.4.3 Design specification	55
2.4.4 Evaluation and results.....	59
2.4.5 Future work.....	60
3 TRUST PLANE	61
3.1 Verifiable Credentials (ATOS).....	61
3.1.1 Overview.....	61
3.1.2 Technology research	62
3.1.3 Design specification	67
3.1.4 Evaluation and results.....	83
3.1.5 Future work.....	83
3.2 Authorization: Network-based Authorization enforcement and authorization distribution (TRU)	84
3.2.1 Overview.....	84
3.2.2 Technology research	85
3.2.3 Design specification	92

3.2.4	Evaluation and results.....	94
3.2.5	Future work.....	94
3.3	Reputation System (UC).....	95
3.3.1	Overview.....	95
3.3.2	Technology research.....	96
3.3.3	Design specification.....	100
3.3.4	Evaluation and results.....	103
3.3.5	Future work.....	104
3.4	Remote Attestation (IPN).....	105
3.4.1	Overview.....	105
3.4.2	Technology research.....	107
3.4.3	Design specification.....	111
3.4.4	Evaluation and results.....	118
3.4.5	Future work.....	118
4	RECOVERY PLANE.....	119
4.1	Self-recovery (XLAB).....	119
4.1.1	Overview.....	119
4.1.2	Technology research.....	120
4.1.3	Design specification.....	121
4.1.4	Evaluation and results.....	123
4.1.5	Future work.....	123
4.2	Credentials recovery (ATOS).....	125
4.2.1	Overview.....	125
4.2.2	Technology research.....	126
4.2.3	Design specification.....	128
4.2.4	Evaluation and results.....	130
4.2.5	Future work.....	130
5	CONCLUSIONS.....	131
	APPENDIXES.....	132
	Appendix A – Analysis of events in Domain A for Reputation System.....	132
	Appendix B – Analysis of events in Domain B for Reputation System.....	133
	Appendix C – Analysis of events in Domain C for Reputation System.....	134
	Appendix D – Policy Manager User Manual.....	135
	Policy Management Dashboard.....	135
	Default policy.....	135
	Updating policies.....	135
	Deleting policies\.....	136
	Adding new policies.....	136
	Policy API.....	136
	Policy fields.....	136
	Backend fields.....	136
	Policy definition fields.....	136
	Endpoints.....	137
	GET /policies.....	137
	POST /policy.....	137
	PATCH /policy/{id}.....	137
	DELETE /policy/{id}.....	137
	GET /default_policy.....	137
	PUT /default_policy.....	137

Appendix E – DID METHODS138

REFERENCES141



LIST OF FIGURES

Figure 1 – ARCADIAN-IoT framework	13
Figure 2 - Sidetree DID Method Overlay network [16]	21
Figure 3 - Public DIDs published on Sidetree Node based on Transmute Sidetree.js Ref [38]	27
Figure 4 - Public DIDs hosted on Service Provider’s DID:WEB endpoint	28
Figure 5 - Privacy preserving peer DID created in SSI wallet per connection	31
Figure 6 - ARCADIAN-IoT Framework Sidetree Node deployment	33
Figure 7 - eSIM component overall view	36
Figure 8 - Open ID connect architecture	38
Figure 9 - Architecture of the Network-based authentication in third-party services	40
Figure 10 - Logical process view of Biometrics component.	48
Figure 11 - Sequence diagram for registration use case.	49
Figure 12 - Sequence diagram for person authentication from a smartphone.	50
Figure 13 - Sequence diagram for person authentication from a video being recorded by drone.	50
Figure 14 - High-level architectural view of the biometrics component and other components that have relation with.	51
Figure 15 – AMQP API specification for biometrics messaging	52
Figure 16 – REST API specification for Biometrics component	52
Figure 17 - ARCADIAN-IoT authentication high-level architecture	56
Figure 18 - Architecture from ARCADIAN-IoT MFA for persons	58
Figure 19 - Authenticated person operation flow	59
Figure 20 - Ledger uSelf built on top of Hyperledger Aries GO Agent	63
Figure 21 - Protocol support for SSI [32]	65
Figure 22 - Cryptographic technology [32]	65
Figure 23 - Register Person in ARCADIAN-IoT Framework by a SP service	68
Figure 24 - Ledger uSelf Broker Self-Sovereign Identity Solution + SSI IdP	69
Figure 25 - Issue a Person VC	71
Figure 26 - Verify a Person VC and provide identity claims	72
Figure 27 - Service Provider service registers a Person	73
Figure 28 - Service Provider deletes a registered Person that it previously registered	74
Figure 29 - SSI IdP Interface description	75

Figure 30 - Self-Sovereign Identity deployment in the ARCADIAN-IoT Framework	76
Figure 31 - SSI IdP Registered Entities	77
Figure 32 - SSI IdP Issuer Screen	78
Figure 33 - QR code display to connect to the SSI Agent	78
Figure 34 - Ledger uSelf mobile SSI Wallet UI	79
Figure 35 - SSI IdP Data Model	82
Figure 36 - 3GPP's PCC Architecture overview ⁴¹	87
Figure 37 - Open5GS architecture	89
Figure 38 - ARCADIAN-IoT Network-based Authorization high-level architecture	92
Figure 39 - Network-based Authorization current technical architecture	93
Figure 40 - Reputation System & Policy Manager logical architecture view	100
Figure 41 - Reputation System internal logic to determine reputation score	101
Figure 42 - OWASP's top 10 IoT security issues (2018 version)	108
Figure 43 - Remote Attestation logical architecture and external dependencies	111
Figure 44 - Remote Attestation for ARCADIAN-IoT (RA2IoT) deployment architecture view	115
Figure 45 - Self-recovery logical architecture view	121
Figure 46 - Recovery of SSI Wallet Credentials Use Case	128
Figure 47 - SSI Credential Recovery Logical Architecture	129
Figure 48 - Screen with default and other policies	135
Figure 49 – Configurable policies fields	136



LIST OF TABLES

Table 1 – Current status of the network-based identification and authentication component	39
Table 2 - Network-based authentication interfaces	40
Table 3: Image data bases	45
Table 4 - Algorithm accuracy	45
Table 5: Machine Learning frameworks	46
Table 6 - MFA interfaces	57
Table 7 - Network-based Authorization component current produced resources	90
Table 8 - Network-based Authorization interface status	92
Table 9 - Data privacy concerns (preliminar analysis)	99
Table 10 - Technologies in the reputation system and policy manager	102
Table 11 - Technologies in the reputation system and policy manager	102
Table 12 - Results of Alpha and Beta testing (with forgetting factor of 0.5)	103
Table 13 DID Method Table [8]	138

ABBREVIATIONS

5GC	5 th Generation Core
5GS	5 th Generation System
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 together 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 2).

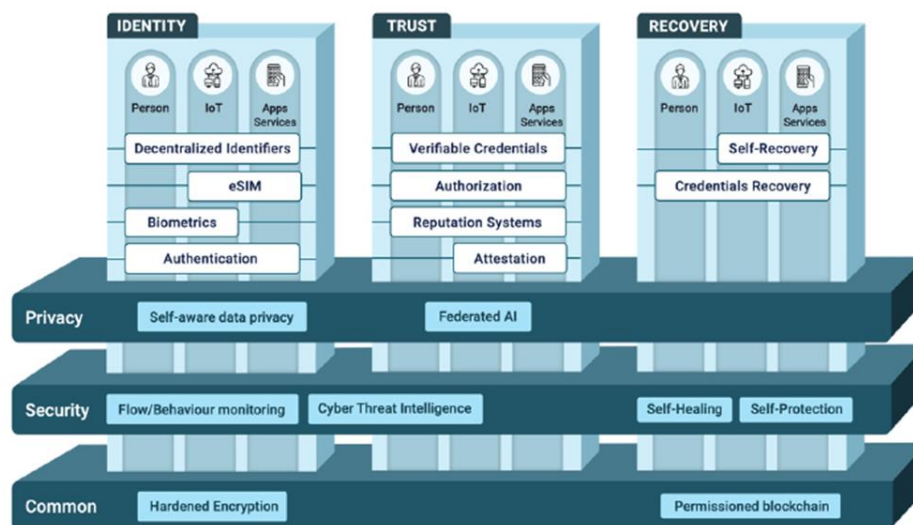


Figure 1 – ARCADIAN-IoT framework

Work Package 4 (WP4) in the ARCADIAN-IoT project is dedicated to the design and technological development of the functionalities that are 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 being conducted from October 2021 to October 2023 and this deliverable (D4.2) details the research activities, the provided resources, and the results of evaluations, that have been obtained within WP4 until December 2022. The reporting takes an agile approach, with D4.2 fully revising D4.1 [46] to provide a self-contained report on the components in the Vertical Plane with this version 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.

¹ https://gitlab.com/arcadian_iot/

- The **Trust plane** implements mechanisms for managing trust on the involved entities (persons, devices and services), namely **Verifiable Credentials** as a method to enable 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 here, with additional details being given in the description of each WP4 component:

- **Objective 1: To create a decentralized framework for IoT systems - ARCADIAN-IoT framework and**
- **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, eSIM and Biometrics)
 - To leverage cellular network authentication processes in a new zero-touch authentication of IoT devices in third-party services
 - 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
- **Objective 4: Provide distributed and autonomous models for trust, security and privacy – enablers of a Chain of Trust (CoT)**
 - 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
- 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 devices
- To support remote attestation involving 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 feed device and service reputation models via Attestation Evidence
- **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

2 IDENTITY PLANE

2.1 Decentralized Identifiers (ATOS)

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:

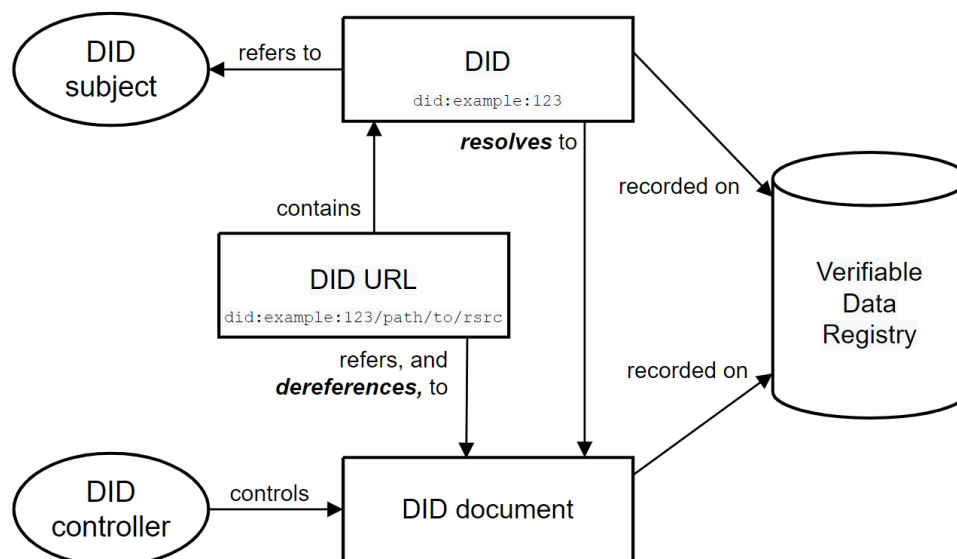


Figure 4 - ARCADIAN-IoT DID solution's basic architecture model [3]

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 under analysis are 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 are 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 is 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 to be deployed in the 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 [1] 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	
Benchmarking (OPTIONAL)	
Not Applicable	
Target value (M30)	Current value (M20)
2	0

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	
Benchmarking (OPTIONAL)	
Not Applicable	
Target value (M30)	Current value (M20)
3	0

KPI scope	
Support, at least two robust identity mechanisms for devices and apps/services.	
Measurable Indicator	
Devices and apps/services support Decentralized Identifiers.	
Benchmarking (OPTIONAL)	
Not Applicable	
Target value (M30)	Current value (M20)
2	0

2.1.2 Technology research

Candidate Verifiable Data Registries (VDRs) under analysis include 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 the 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 by different means, from decentralised and distributed to federated and centralised.

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 repository², 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 online³, 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 Infrastructure⁴ (EBSI) and within that initiative the European Self-Sovereign Identity Framework⁵ (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.

² <https://github.com/decentralized-identity/Sidetree>

³ <https://github.com/decentralized-identity/ion-tools#ionjs>

⁴ <https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Home>

⁵ <https://decentralized-id.com/government/europe/eSSIF>

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 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 Integration with Permissioned Blockchain to provide a trust anchor for publishing the DID Doc

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 2 shown below.

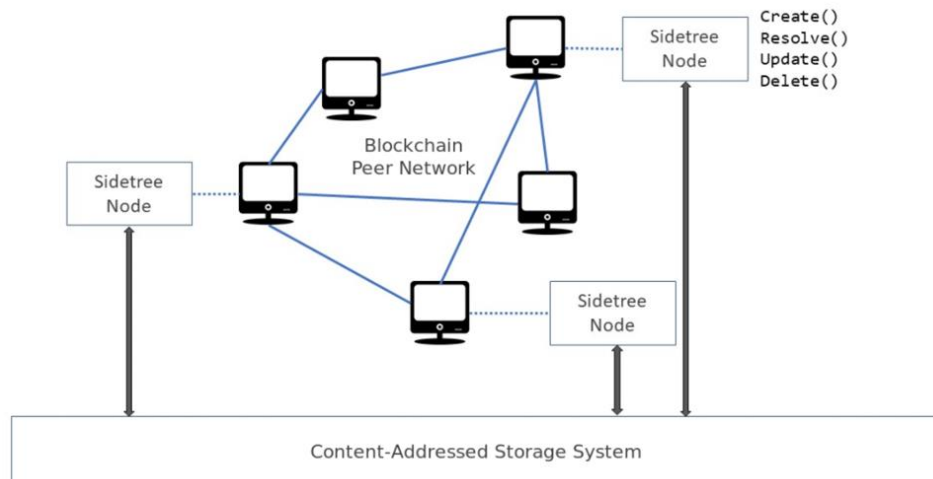


Figure 2 - Sidetree DID Method Overlay network [16]

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 *Integration with 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.

2.1.2.1.4 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 ed25519S and EcdsaSecp256k1 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.

This functionality is planned for the second prototype and therefore not detailed in design specification of section 2.1.3.

2.1.2.1.5 DID Authentication

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 all entity (Persons, IoT Devices, Services) identities in the ARCADIAN-IoT framework is aimed at being provided by Verifiable Credentials (see section 3.1).

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

That said, where Verifiable Credentials are not found to be not suitable e.g. for constrained devices, then alternatives based on “DID authentication” are proposed also 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] is 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.

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.

As a result it is 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. This would be suitable for IoT Devices and SSI Issuers and the ARCADIAN-IoT Framework SSI Agent that can benefit from public DIDs.

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 and 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.

It is also identified above the use of BBS+ signatures as the ideal way to support ZKP privacy preserving measures.

2.1.2.3 Produced resources

ATOS have implemented a sidetree node integrated with private Ethereum and hosted by ATOS for integration with the first prototype.

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
Note: The IoT Device DID operation will be supported in the final prototype P2.

2.1.3.1.2 ARCADIAN-IoT Framework SSI Agent´s DID Management

The creation and management of public DIDs for the ARCADIAN-IoT SSI Agent, supported by blockchain.

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.

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 the ARCADIAN-IoT framework. This is handled out-of-band where a SP administrator will email to ARCADIAN-IoT administration their request to add their public Decentralized Identity to the ARCADIAN-IoT framework.

2.1.3.1.7 Register Service Provider Service to the ARCADIAN-IoT

Once an SP has been registered in ACADIAN-IoT the SP can automatically register their services in the framework with a public DID associated to the service.

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. This decision may also be taken by the ARCADIAN-IoT administration. All SP Services must first be deleted before the SP can be deleted.

2.1.3.1.9 Delete a registered Service Provider Service to the ARCADIAN-IoT

Once an SP has been registered in ARCADIAN-IoT the SP can automatically delete their register services in the framework with a public DID associated to the service. If there still exists registered users or IoT-Devices for the service, then it cannot be deleted.

2.1.3.2 Logical architecture view

2.1.3.2.1 Public DID published on Sidetree for IOT Devices & ARCADIAN-IoT Agent

To support public decentralized identifiers in ARCADIAN-IoT framework as discussed in section 2.1.2.1.2 the sidetree specification is implemented 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.

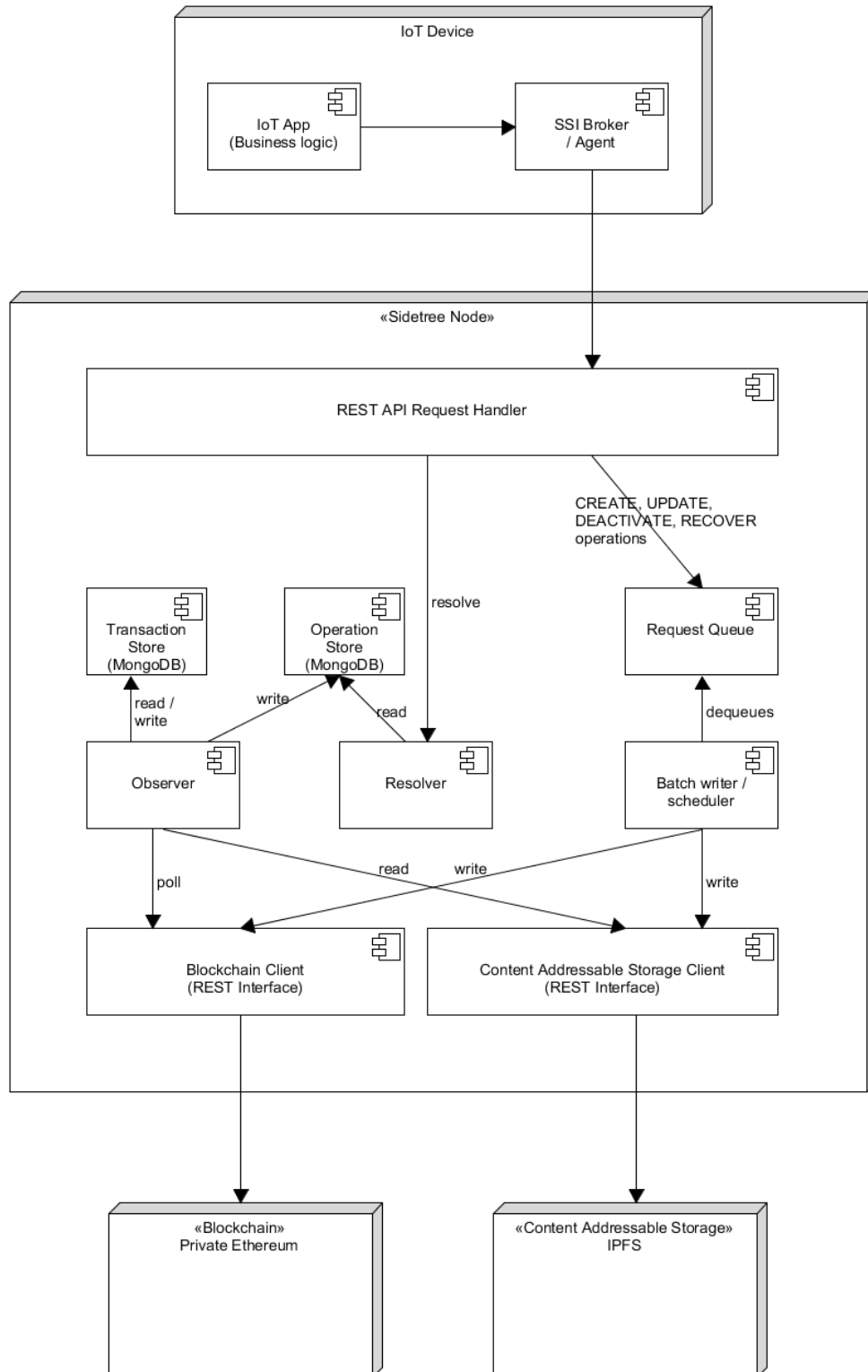


Figure 3 - Public DIDs published on Sidetree Node based on Transmute Sidetree.js Ref [38]

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 published on DID Web for Service Provider authentication

To support Service Provider (SP) onboarding to the ARCADIAN-IoT framework and authentication to the framework’s services each SP organisation will host their on DID Doc as per the DID:WEB specification presented in section 2.1.2.1.1.

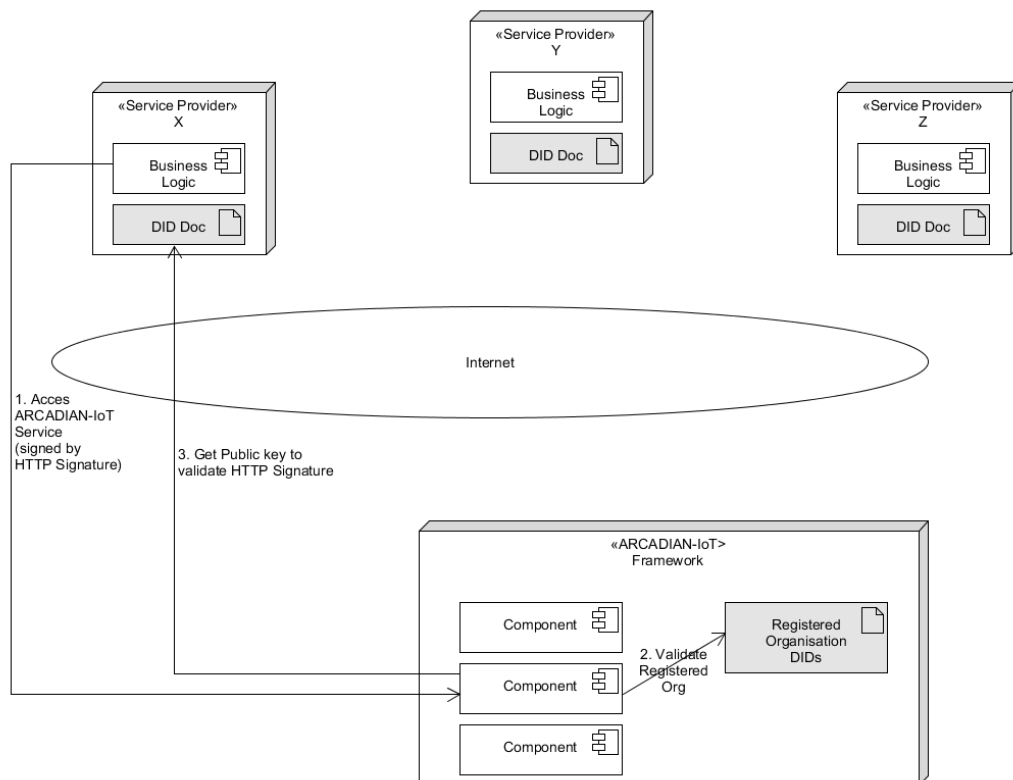


Figure 4 - Public DIDs hosted on Service Provider’s DID:WEB endpoint

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. 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/.well-known/did.json

As long as the SP is registered, and the API call signature is validated it will be considered authenticated and authorised to access the ARCADIAN-IoT service.

Additionally, each SP service will 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, noting that currently only the RSA key verification is supported for DID WEB:

```
{
  "@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_CJEwxMhZgwggitO6
HPiaqk",
      "controller": "did:web:example.com:service:drone_buddy",
      "publicKeyPem":
"MIIBljANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxZ3Sjwi2hdw80jQO8f/jTPMjtcJU
imRR8wHkEPayQ+0rcCxmV87xK8kyemr1kns3A6ow8Pf1L0mzBmX4XWpzGPkEtRISBz/l6IC
EwdWK/QQSHirDFSGF8Bo6C8EnN+Cwq5ck+Vbr2hzNfY6LQmuy2hVl5EYLuesMQRd5IBSB
LVkyLdlcwrQUUKfT1kxPXS2ILG5GtVU6sWSngZl8JIQLg7pbuzzugCgKVjgmtkWwoTiqbFs7jY
```

```

8P0XZ233tXeG/KMDMZsPIRdSIBzq5zOBCEnpFzTpbPXUD7VaEXHOS8Ox6EYCuJG2Lt6n
+iC8qe5mOH+NU5Wjd1pjipLX1SFkMmTQIDAQAB"

    }

  ],

  "authentication": [

    "did:web:example.com:service:drone_buddy#ECEA8_A1SlddIICzj4SFS_CJEwxMhZgwgitO6
    HPiaqk"

  ],

  "assertionMethod": [

    "did:web:example.com:service:drone_buddy#ECEA8_A1SlddIICzj4SFS_CJEwxMhZgwgitO6
    HPiaqk"

  ]

}

}

```

The API call signature makes use of HTTP Signatures specification [39] to authenticate the API. HTTP Signatures enables 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 through 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.

2.1.3.2.3 *Privacy preserving Peer DID for Mobile Wallets*

Peer DID conform to the W3C Decentralized Identity specification [40] and are 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 will establish connections to entities in a pair-wise fashion as per the figure below.

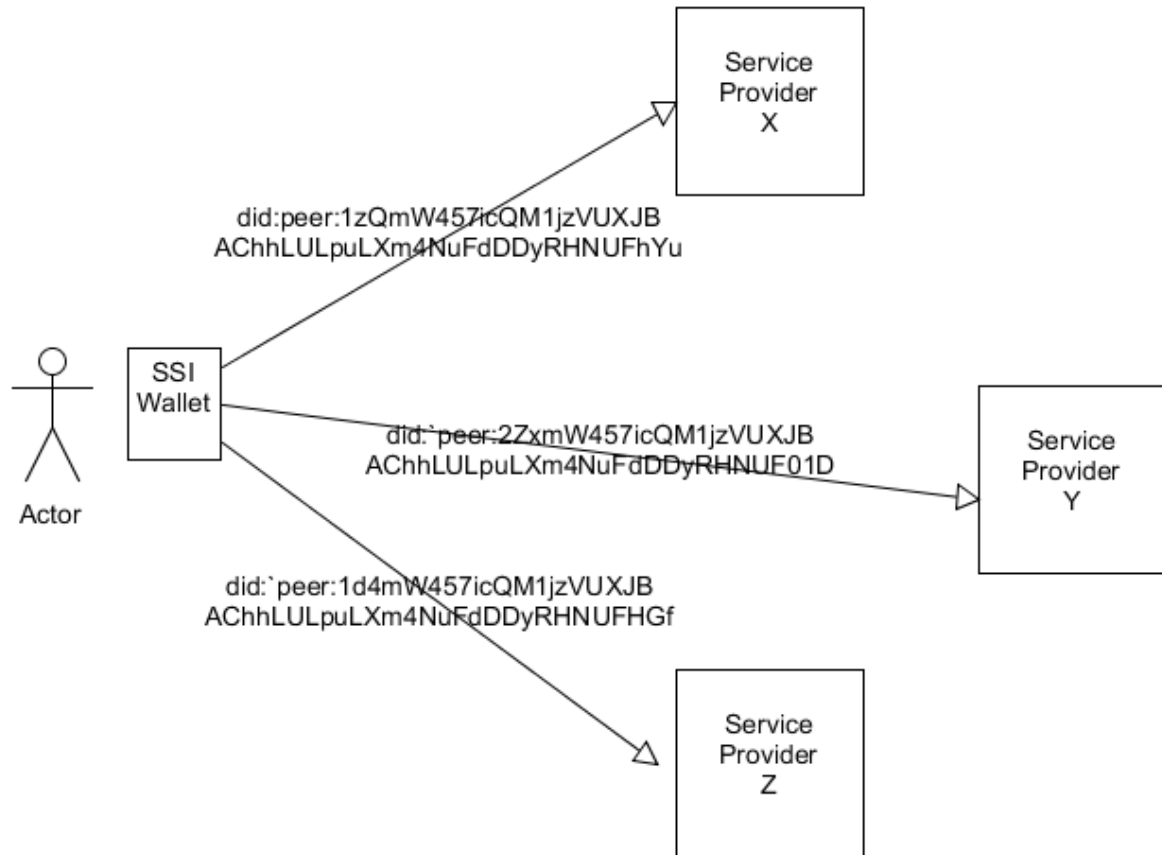


Figure 5 - Privacy preserving peer DID created in SSI wallet per connection

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 Sidetree

Public DID operations are fully implemented and handled by the initial setup and configuration SSI Agents and their interaction with a Sidetree Node. As both components are provided by ATOS the sequence diagram is not fully needed to show integration with other partner components. The SSI Agents will be configured upon initial restart to request a public DID from a Sidetree Node.

Sequence diagrams involving the SSI Agent and facilitated by the public DID can be found in section 3.1.

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 Sidetree Interface

The Sidetree Node supports the DID Operations specified in the sub-use cases section 2.1.3.1.

The CREATE DID Operation is supported in the first prototype and is called with the public keys needed to be published with the Decentralized Identity.

Below is a non-normative example of the public keys provided in the CREATE DID Operation:

```
{
  "@context": "https://w3id.org/did/v1",
  "publicKey": [
    {
      "id": "#primary",
      "usage": "signing",
      "type": "Ed25519VerificationKey2018",
      "publicKeyJwk": {
        "kty": "OKP",
        "crv": "Ed25519",
        "x": "PC20Ganm5IKCJffoDV7zCK2_LyLrDMWOKy43HXISWcQ",
        "d": "sjDaMXAeGSJTUJBwP7Ft36Yc7GF93Ee3Cjw-9Go8zhc",
        "kid": "OtZJamqSEzJLSAAEcVK3Un0x6C7sNb7tbpBoxim1hIQ",
        "use": "sig"
      }
    },
    {
      "id": "#recovery",
      "usage": "recovery",
      "type": "Ed25519VerificationKey2018",
      "publicKeyJwk": {
        "kty": "OKP",
        "crv": "Ed25519",
        "x": "OL20Ganm5IKCJffoDV708K2_LyLrDMWOKy43HXISW56",
        "d": "xxDaMXAeGSJTUJBwP7Ft9yYc7GF93Ee3Cjw-9Go8zbd",
        "kid": "34ZJamqSEzJLSAAEcVK3UPvx6C7sNb7tbpBoxim1h89G",
        "use": "sig"
      }
    }
  ]
}
```

2.1.3.4.2 HTTP Signature Interface

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.

2.1.3.5 Technical solution

2.1.3.5.1 Deployment architecture view

Sidetree node deployment:

A Sidetree Node is deployed in the ARCADIAN-IoT Framework and integrates with a Private Ethereum Network. The SSI Agent residing in the IoT Devices will implement the interface to the Sidetree node to create a DID over the sidetree node upon initialisation of the device.

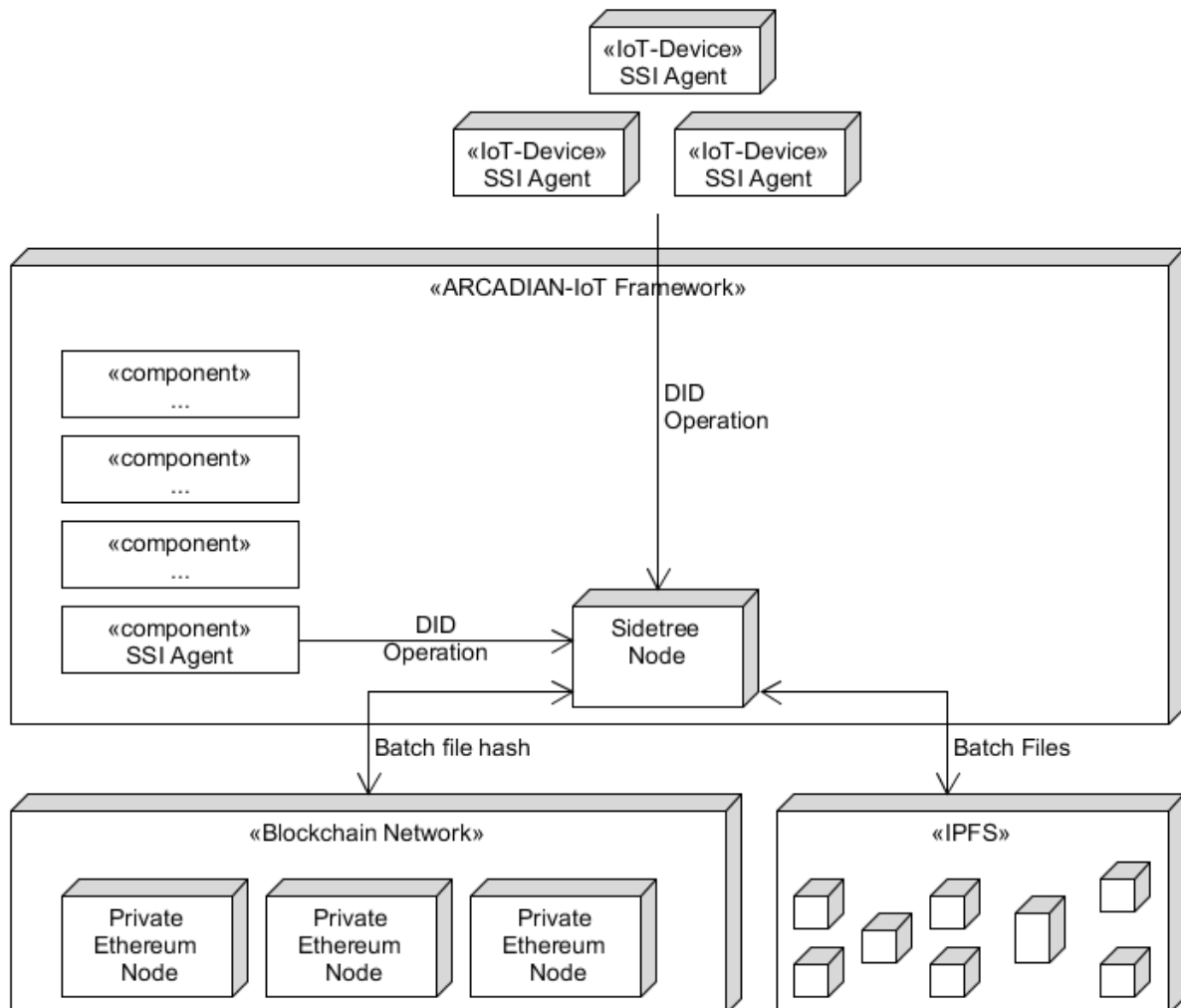


Figure 6 - ARCADIAN-IoT Framework Sidetree Node deployment

2.1.3.5.2 API specification

The Sidetree DID Operation APIs are called by the SSI Agent software provided by ATOS.

2.1.4 Evaluation and results

The results for the first prototype are to provide:

- A Decentralized Identifier created by the Sidetree Node deployed in local test environment can be resolved here: <https://resolver.prod.ari-bip.eu/1.0/identifiers/did:elem:uself:EiCx0ljXVtxuFmVFIUEgifx8rRLBjwmZ0Zy-5xtLuWgsOw>
- A Decentralized Identifier based on DID:WEB is resolved here: <https://resolver.prod.ari-bip.eu/1.0/identifiers/did:web:uself.prod.ari-bip.eu:testing:test>

The resolved DIDs available on the URL links are validated against the W3C Decentralized Identifier specification [3].

2.1.5 Future work

The Sidetree Node is currently under integration testing with the ARCADIAN-IoT Framework's SSI Agent for creating a public Decentralized Identifier. Once complete, this will support the initial prototype.

The Sidetree Node has also been integrated recently with a private instance of Ethereum and not with Hyperledger Fabric which is the permissioned blockchain technology selected for the ARCADIAN-IoT framework. It will be explored the integration of Sidetree with Hyperledger Fabric in Task 3.1. It will be also further investigated an alternative to support Decentralized Identifiers on the Permissioned Blockchain making use of the Publisher Smart Contract developed for ARCADIAN-IoT Framework in Task 3.1.

The implementation of BBS+ signatures will be pursued to provide ZKP privacy preserving measures.

The final work is to then continue to deploy in the ARCADIAN-IoT Framework for the piloting sites for the final prototype.

2.2 eSIM – Hardware-based identification and authentication (TRU)

2.2.1 Overview

2.2.1.1 Description

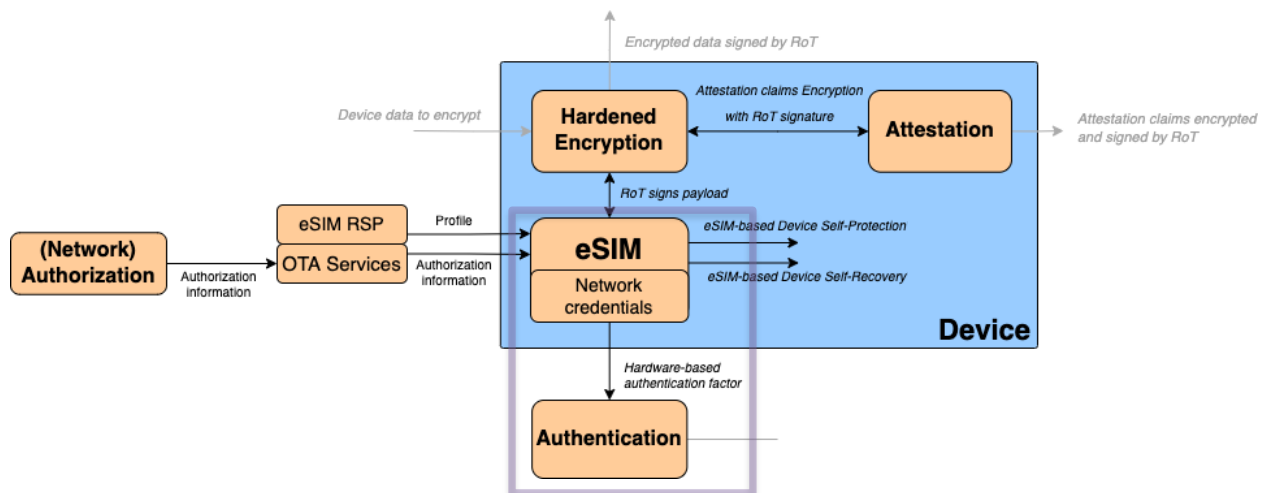
The eSIM is the evolution of the well-accepted and widespread SIM card technology to an embedded format with remote provisioning and management capabilities, while maintaining its security processor characteristics. Its ecosystem provides a fully digital management of devices' connectivity and is in itself an enabler of innovation in terms of potential for automation (e.g. for provisioning and management of a large number of devices connectivity 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 will act 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 will be, therefore, a relevant security agent for connected devices with several roles as depicted in

Figure 7. In this section, which belongs to identity management, the report of the ongoing eSIM-related work will focus on its role related with the item b. above.

Figure 7 - eSIM component overall view⁶

ARCADIAN-IoT authentication will rely on a multi-factor process to identify and authenticate users and devices (section 2.4). The **network-based authentication**, focused in this section, is one of these factors and it presents the framework with a novel method to authenticate an eSIM-equipped device in a third-party service by leveraging cellular networks authentication⁷, whose credentials and processes are securely stored at hardware level in the device eUICC.

The network-based authentication component leverages the standardized and widely used authentication mechanism of cellular networks, 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 cyphering⁸. These well-accepted security processes rely on SIM technologies to have stored the information and processes necessary, in a hardware secure element.

By leveraging this mechanism, the network-based authentication component described in this section aims to extend the well-accepted standard of authentication of devices in cellular networks (state-of-the-art), to a new form of identification and authentication of devices and persons in third-party services (beyond state-of-the-art). Regarding persons, we envision that their identification will be attached to the one of their personal devices (stored in the secure element of the personal devices), and that the attachment process (person to personal device) will happen in the registration moment.

TRU already has a patented⁹ experimental proof of concept of this technology in TRL 3 (TRL at the beginning of the project) and aims to bring it to TRL 6 within the context of ARCADIAN-IoT, researching to enhance and demonstrate it in at least two IoT domains targeted in the project. Another objective is to make this technology agnostic to the IoT devices characteristics in terms of processing power, energy consumption and communication protocols used (e.g. being a technology ready for IoT use cases with high computing power demands; and ready to IoT use cases with high constraints of energy, computing or communication (e.g. a device whose battery needs to last for 10 years, with computing power just to read a sensor and send it to a cloud provider once a week).

⁶ Diagram from D2.5 depicting eSIM multiple roles in ARCADIAN-IoT

⁷ 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

⁸ 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))

⁹ https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2021224624&_cid=P10-L12DVI-41405-1

2.2.1.2 Requirements

The requirements¹⁰ for the network-based authentication component are the following:

- Leverage cellular network authentication procedures to **authenticate devices and persons in 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 is that ARCADIAN-IoT's IoT and personal devices that will make use of the network-based authentication component need to **support eSIM** (or, alternatively, any other form of SIM).

2.2.1.3 Objectives and KPIs

The network-based identification and authentication in third-party services contribute to the accomplishment of the following objectives and KPIs.

KPI scope	
In the context of the objectives of <i>enabling security and trust in the management of objects and persons' identification</i> , we aim to use eSIM to support an identity approach at hardware level, which should be a robust identity mechanism for devices; and <i>enable an identification approach for persons</i> to be joint with others for a set of 3 simultaneous identification approaches for persons. In this sense, the scope of this component focuses on leveraging cellular authentication processes, where subscribers' identity and identification process is securely stored in the UICC/eUICC, to perform a zero-touch authentication of IoT devices and persons in third-party services.	
Measurable Indicator	
1. Leverage cellular network authentication processes in a new zero-touch authentication of IoT devices in third-party services (Y/N) 2. Number of different devices where the innovation is demonstrated 3. TRL	
Target value (M30)	Current value (M20)
1. Y 2. At least 2 (1 IoT device and 1 personal device) 3. 6 (prototype demonstrated in relevant environment)	1. Y 2. 1 3. 4 (component validation in laboratory)

2.2.2 Technology research

TRU's patent previously mentioned refers to a technology in which the device that wants to authenticate needs to request a signed token from a core network service for that purpose. This core network service would verify the device identity and authentication status, meaning, if it is a known subscriber that already authenticated to the network and, if so, it would issue a signed and protected token (JWT) to the device so it could use it in the third-party it wanted to authenticate to.

While functional, this process is demanding for some IoT devices, e.g. constrained devices with battery that needs to last for years. Apart from the previous process, devices would also need to use the right protocol to communicate with the (or with each) third party. The research in the current period focused on developing a solution that removed the aforementioned burden to IoT

¹⁰ Requirements were updated from the ones described in previous deliverables to better fit the component objectives

devices as possible, aiming for a technology as agnostic as possible to the devices' characteristics and communication protocols.

2.2.2.1 Background

Open ID Connect¹¹ (Figure 8) inspired many of the work done in this component research. In this protocol there are three key actors: (1) the Service Provider that provides a service to clients but does not authenticate them directly, (2) the Client that needs to be authenticated to access the Service Provider and (3) the ID provider that can authenticate clients on behalf of the Service Provider.

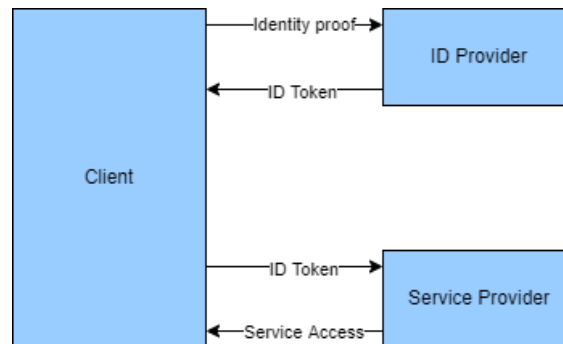


Figure 8 - Open ID connect architecture

The protocol starts when the client authenticates itself to the ID provider. This provider then issues an ID token that proves the client identity. The client then uses this ID token to authenticate itself to the Service Provider that can then provide or deny access to its service based on this authentication.

In our research, considering that the ID Provider is the network provider, we are extending this concept, studying the impact of avoiding communication flows, which can enhance energy-related matters in IoT devices.

2.2.2.2 Research findings and achievements

The research performed allowed to find what seems to be a suitable technology for leveraging cellular network authentication to authenticate cellular devices to third party services, optimized for being more agnostic to the device or communication protocol.

Departing from the Open ID connect flow (Figure 8), the current hypothesis/approach is to place the ID Provider between the client and the service provider, to whom the client wants to authenticate to. Figure 9 depicts exactly this with the *Notarizer* (ID provider) placed between the device and the service provider. This reduces the number of flows that the device needs to execute and allows to have communication intelligence in the core network function, meaning, to adapt the communication protocol to what is requested by the third party in a network function, removing thus the burden of having more than one communication protocol configured at devices if they need to communicate to more than one third-party. This hypothesis doesn't add any shortcoming, because the core infrastructure of the network would assume these two roles in any of the cases: the ID provider role; and the means for the devices to reach the internet services (and the Service Provider).

The unique flow needed in the current approach is the authentication request from the device, informing the third-party it wishes to authenticate to. The core network component verifies the

¹¹ [Final: OpenID Connect Core 1.0 incorporating errata set 1](#)

identity proof and, if valid, it generates and appends the ID token to the authentication request and forwards it to the third-party service.

The service provider (or ARCADIAN-IoT framework as will be seen in the multi-factor authentication section) will need to have a component able of verifying the validity of the ID token provided (e.g. validating the signature of the ID provider).

Table 1 depicts the current achievements in terms of the status of the prototyping of the described technology.

2.2.2.3 Produced resources

The technology produced so far was described in the previous section and is specified in Table 1.

Table 1 – Current status of the network-based identification and authentication component

#	Subcomponent	Brief description	Prototyping status
1	Notarizer	Validates in the network core a device identity as being from a known network subscriber with successful GSM authentication. If successful, this component generates and appends an ID token to the authentication request and forwards it to the target third-party services	The current prototype has the functionality described before. Functional testing in lab environment with a real device was successful.
2	Network-based authenticator	Confirms the validity of the provided ID token	Prototype with successful functional testing.

The above-mentioned subcomponent prototypes were developed using standard technologies (JWT¹² for the ID token, and developed in Rust / GO for the *Notarizer* – both languages were used for potential future comparison in what concerns performance and scalability).

2.2.3 Design specification

2.2.3.1 Logical architecture view

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

Looking at Figure 9, we start by assuming that the device has a valid ARCADIAN-IoT eSIM provisioned and active (subscriber valid in Truphone's network), and it is already authenticated and attached to the cellular network according to the regular and well-accepted GSM process. After, the beyond state-of-the-art authentication process starts when the device needs to send a message to the third-party (IoT service provider) it intends to authenticate to. The device communicates with the *Notarizer*, subcomponent that is positioned in the core network, sending it the authentication request with the intended destination. The *Notarizer* uses the device's network identifiers (previously used to authenticate in the network) to confirm its identity in a subscriber database at core network infrastructure. In case of validation success, the *Notarizer* crafts a *Network ID Token*, which allows to identify and authenticate, with the same level of security and trust of the cellular network authentication, the network subscriber, or the device in this case, at the compliant third-party. This *Network ID Token* 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. This authenticator validates the token and sends the result to

¹² <https://www.rfc-editor.org/rfc/rfc7519>

ARCADIAN-IoT *multi-factor authentication*, to join 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 ID token is generated and returned to the IoT service provider, who returns it to the device for its normal authenticated operation.

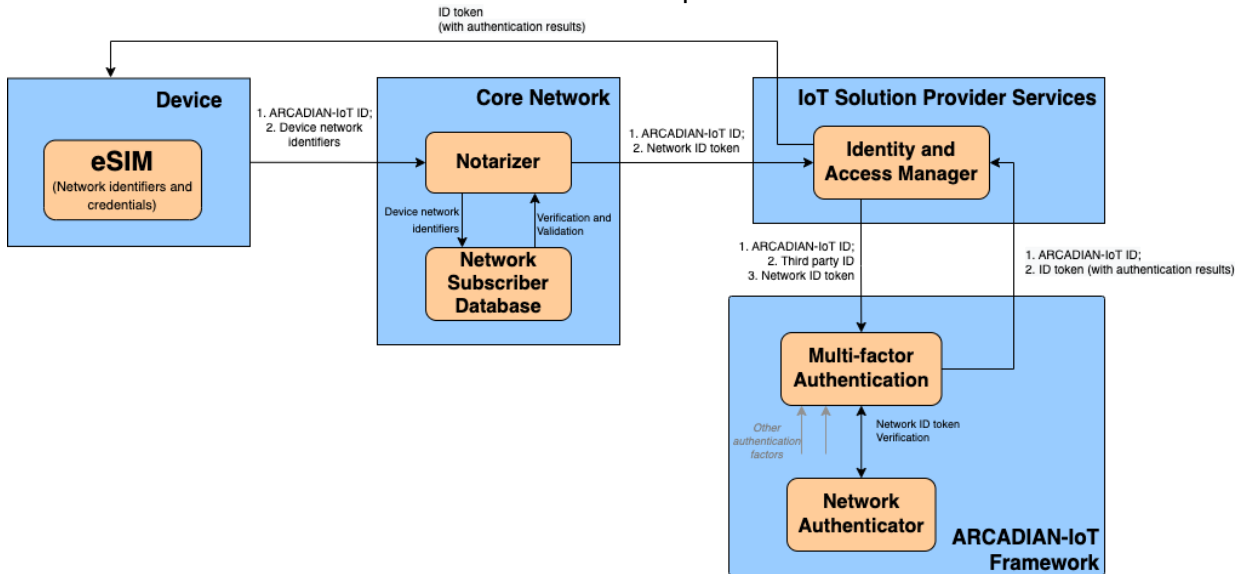


Figure 9 - Architecture of the Network-based authentication in third-party services

2.2.3.2 Interface description

In terms of interfaces, the network-based authentication subcomponents, 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 Table 2.

Table 2 - Network-based authentication interfaces

Sender	Receiver	Communication type	Content exchanged	Status
Device (IoT or personal) service	Notarizer (Network-authentication subcomponent)	RESTful API	Authentication request	Done for P1
Notarizer	IoT service provider services	RESTful API	Authentication request w/ network ID token	Done for P1
Multi-factor Authentication	Network authenticator (Network-authentication subcomponent)	RESTful API	Network ID token + Information about its validity	Done for P1

2.2.3.3 Technical solution

By leveraging the authentication mechanism already present in the cellular network, it is possible to extend it by using the network operator to issue ID tokens, essentially extending the cellular network authentication mechanism to an ID provider, capable of exporting its authentication to any Service Provider that can trust the claims made by the cellular authentication service.

This technology is based on two widely used authentication mechanisms. Open ID Connect is considered safe and reliable, and the GSM cellular authentication, which is widely well accepted. The current solution is inspired in a patented work from Truphone¹³, which has a proof of concept, considered to be in TRL 3 at the beginning of ARCADIAN-IoT project. With the research done in we expect to take it to TRL 6, with its demonstration in three relevant IoT domains. One of the identified areas of improvement is the solution adaptability to different IoT device characteristics, namely energy constraints, research that is ongoing.

In ARCADIAN-IoT the network-based authentication will be integrated in a multi-factor authentication schema, which will be described in the authentication section.

2.2.4 Evaluation and results

The current solution, particularly the core network function (*Notarizer*), is already working with real devices (real network subscribers). The solution has been functionally tested using a Raspberry Pi 4 Model B, connected to a Monarch GM01Q Module LTE Category M1 Evaluation Kit. Test service provider services were also created for testing the solution end-to-end. Within this lab setup the system functionally behaved as expected. The current solution, by reducing the number of flows of OIDC, and allowing to adapt the communication protocol between the device and the IoT service provider in a core network function, is more fit to constrained IoT devices.

Testing and evaluation efforts with two domain owners (real IoT service providers) is ongoing in the scope of WP5.

2.2.5 Future work

According to the current research, the current solution seems to fulfil most of the KPIs defined to this component. The focus on the next deliverable will be on increasing its TRL, mainly with its integration and demonstration in at least 2 IoT service providers' services (and the existent devices). Therefore, TRU will be working with the IoT service providers to gather inputs for the enhancement of the component. Also, the integration with ARCADIAN-IoT's onboarding process is ongoing and is part of the future work.

2.3 Biometrics (UWS)

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 from the frontal camera of a smartphone in order to start the smartphone app as part of the multi-factor authentication. Second, a more challenging scenario is the face verification in a video recorded from an Unmanned Aerial Vehicle (UAV). In this scenario the biometrics component will support 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 will rely on a multi-factor authentication process to identify the users that requests a particular service. This component will authenticate a person from different distances.

¹³ https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2021224624&_cid=P10-L12DVI-41405-1

2.3.1.1 Description

The UWS Biometric component will be 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 step. 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 starts 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, we will explore the possibility of providing together with the identification result, the bounding box that represents the part of the scene where the face has been identified to allow a GUI to create an overlay figure.

2.3.1.2 Requirements

After further analysis, the requirements specified in previous deliverable 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 in a user registration phase.
- 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 the reception of an image from the frontal camera of the user's smartphone to authenticate the user.
- 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)	
Benchmarking (OPTIONAL)	
8 FPS	
Target value (M30)	Current value (M20)
16 FPS	16 FPS

KPI scope	
End-to-End speed of the biometrics process.	
Measurable Indicator	
Frames per Second (FPS)	
Benchmarking (OPTIONAL)	
1.5 FPS	
Target value (M30)	Current value (M20)
5 FPS	1.5 FPS

KPI scope	
High accuracy of the face verification algorithm at close distances (less than 2 meters).	
Measurable Indicator	
F1 score or mAP.	
Benchmarking (OPTIONAL)	
89.05% mAP	
Target value (M30)	Current value (M20)
Over 90% mAP	89.05% mAP

KPI scope	
High accuracy of the face verification algorithm at far distances (more than 2 meters).	
Measurable Indicator	
F1 score or mAP.	
Benchmarking (OPTIONAL)	
65.23% mAP	
Target value (M30)	Current value (M20)
Over 70% mAP	68.23% mAP

KPI scope	
Reliable verification of the face at close distances (less than 2 meters).	
Measurable Indicator	
False Acceptance Rate (FAR)	
Benchmarking (OPTIONAL)	
0.5% FAR	
Target value (M30)	Current value (M20)
Below 0.5% FAR	0.5% 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)	
0.5% FAR	
Target value (M30)	Current value (M20)
Below 0.5% FAR	0.5% FAR

KPI scope	
Cost-effective camera and drone platform (hardware only).	
Measurable Indicator	
Euros (€)	
Benchmarking (OPTIONAL)	
N/A	
Target value (M30)	Current value (M20)
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, algorithm 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
DroneSURF ¹⁴	441,000	58 people	Yes	Missing different lightning conditions and distance classification.

¹⁴ 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.

LFW¹⁵	13,233	5749 people	No	Close range images.
Color FERET¹⁶	14,126	1199 people	No	Close range images.

Table 3: Image data bases

2) Algorithm

Once the database is collected, it needs to be trained with an 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)
ArcFace¹⁷	50	99.84%	64.93%
VGG-Face¹⁸	110	98.95%	74.67%
Dlib¹⁹	10	99.38%	60.54%

Table 4 - 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)
TensorFlow²⁰	CPU / GPU	Open Source	Medium	High

¹⁵ 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*.

¹⁶ 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.

¹⁷ 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).

¹⁸ Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep face recognition.

¹⁹ King, D. E. (2009). Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10, 1755-1758.

²⁰ Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens,

PyTorch ²¹	CPU / GPU	Open Source	Medium	Medium
OpenCV ²²	CPU	Open Source	Easy	Medium
SNPE ²³	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 identification 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 is needed in the face identification algorithm. This produced result is also exposed in the following section.

In order to execute the face identification 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 event 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

Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng.

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

²¹ Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (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.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>

²² Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

²³ Snapdragon Neural Processing Engine SDK, SNPE (2020). <https://developer.qualcomm.com/docs/snpe/overview.html>.

and distances from the faces recorded. Besides, this dataset will convert a wide range of different scenarios in low-lighting 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 continuously evolving process as new data is being added periodically.

Dataset	Size (images)	Identities	Drone Friendly	Notes
UWS Dataset	7167	20	Yes	Currently different conditions. missing lighting

Table 7: Dataset specification

- Algorithm

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 62.5 ms and 68.23% mAP of accuracy at far distances. UWS Model is 3% more accurate at far distances than standard ArcFace while maintaining the same speed.

Algorithm	Speed (ms)	Accuracy	Far Distance Performance
UWS Model	62.5 ms	68.23%	Improved performance.

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 14 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 10 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 RetinaFace 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 to determine if the distance of both embeddings is close enough to verify the identity of the person that requested by the authentication service.

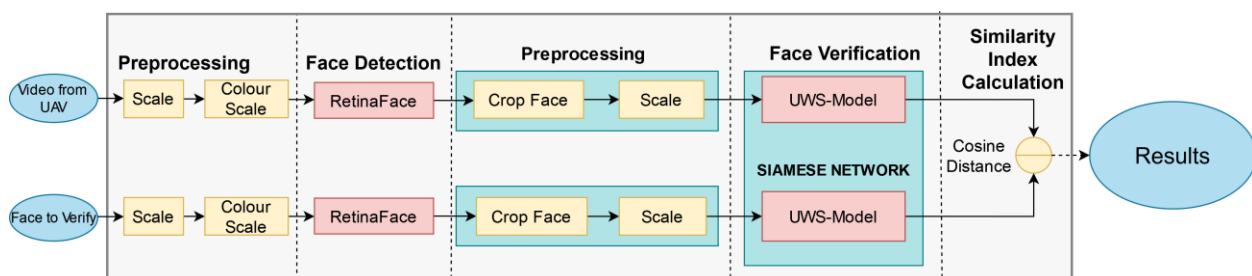


Figure 10 - Logical process view of Biometrics component.

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. Third-Party service is the only entity that intervenes 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 is 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 11, 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.

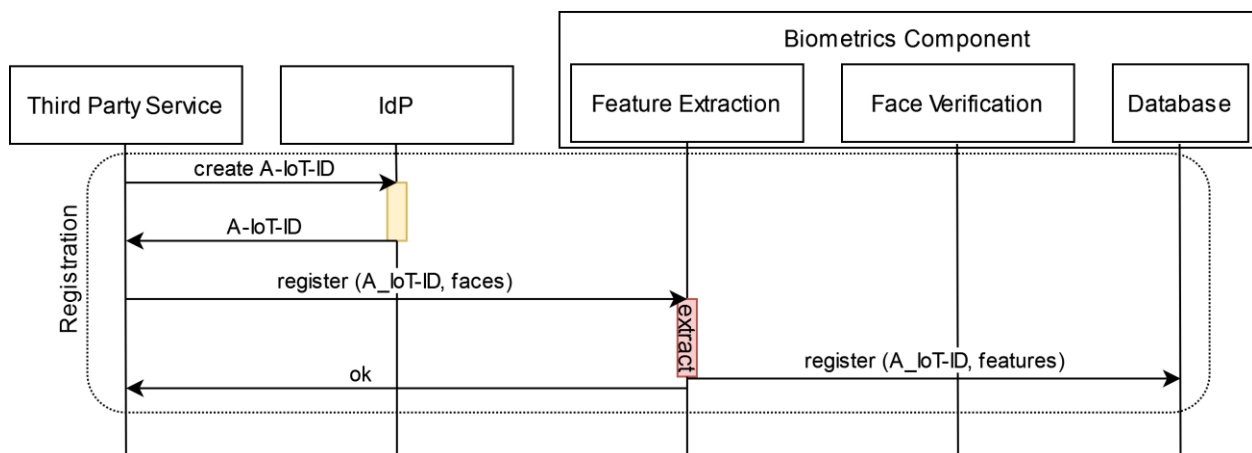


Figure 11 - Sequence diagram for registration use case.

Authentication from smartphone is presented in Figure 12. 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.

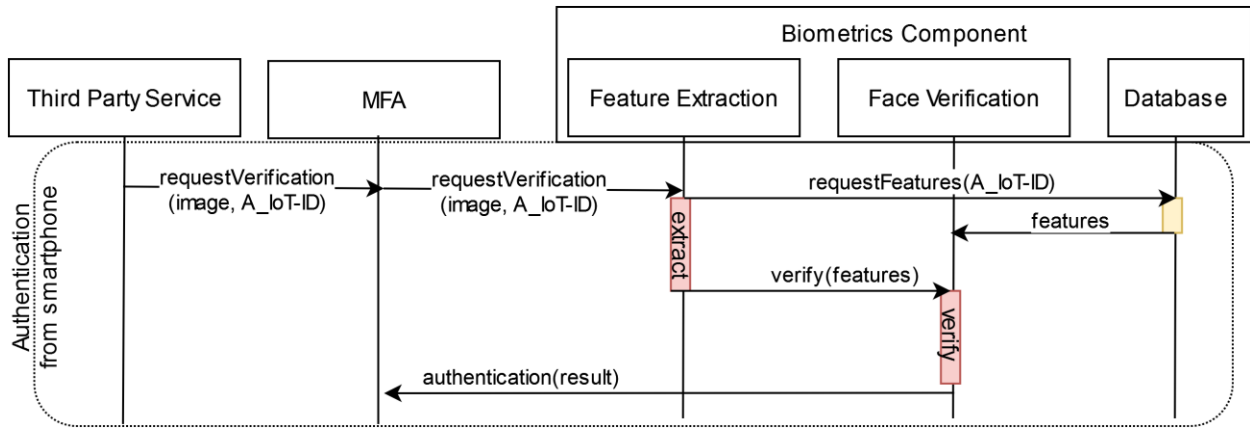


Figure 12 - Sequence diagram for person authentication from a smartphone.

Authentication from a video recorded by a drone is presented in Figure 13. 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.

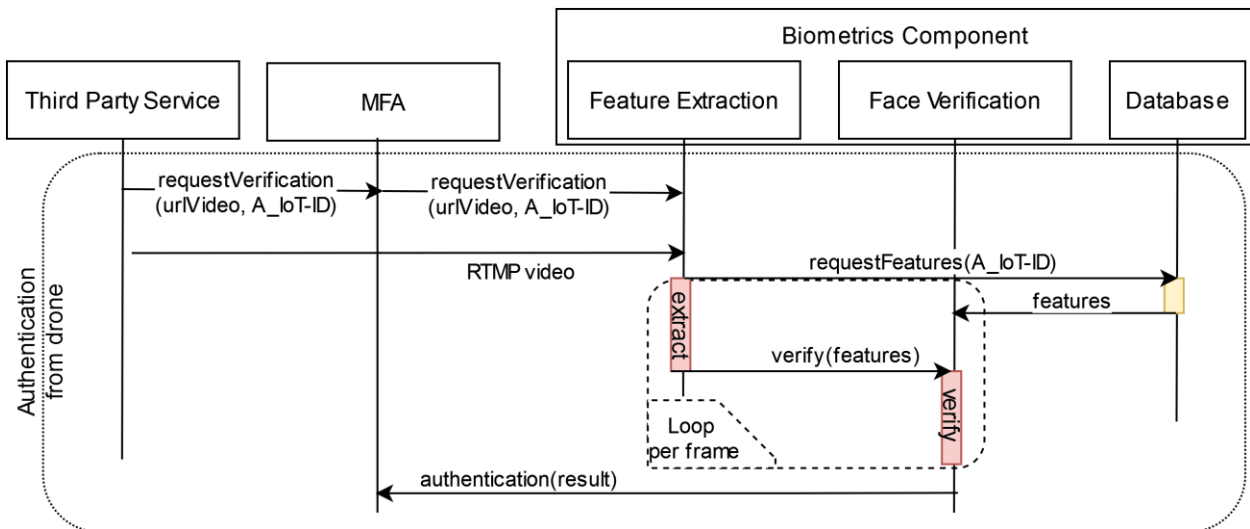


Figure 13 - Sequence diagram for person authentication from a video being recorded by drone.

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 (in this case Domain A, led by LOAD). 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 14 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.

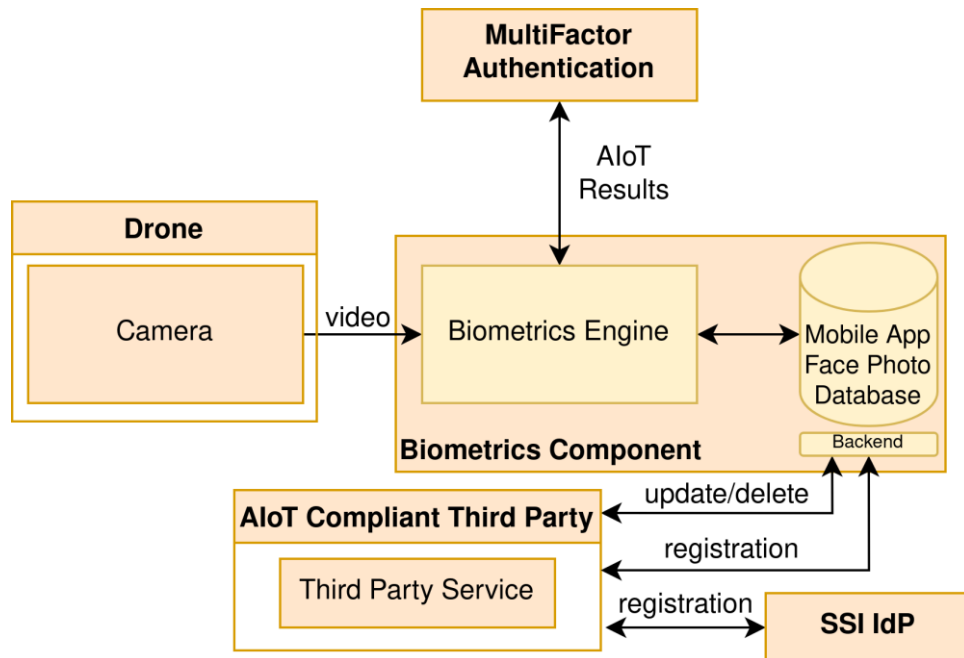


Figure 14 - High-level architectural view of the biometrics component and other components that have relation with.

2.3.3.5.2 API specification

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

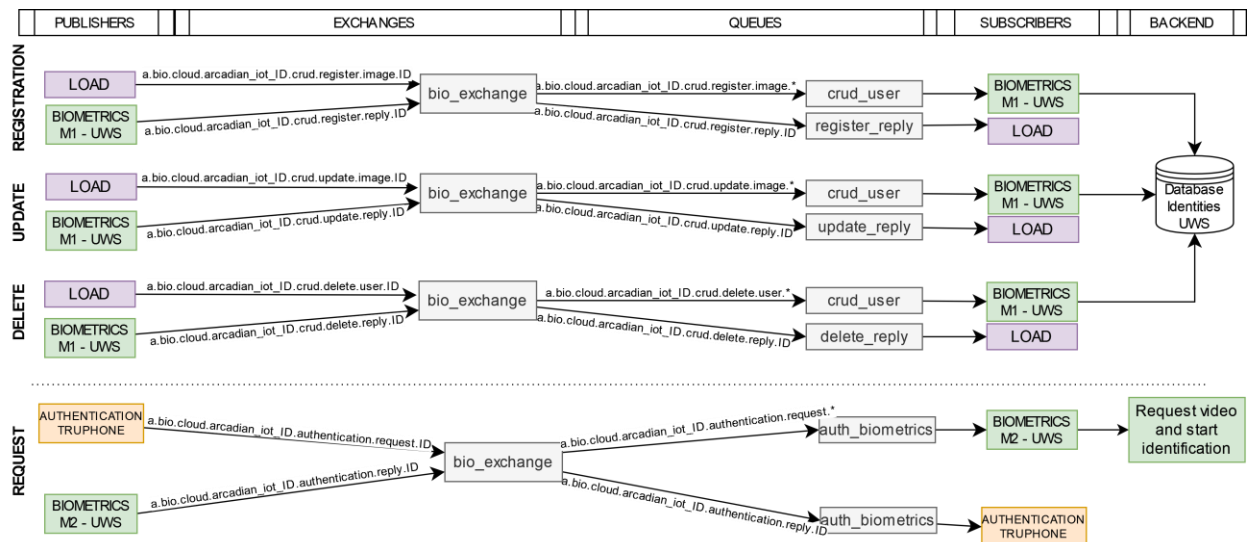


Figure 15 – AMQP API specification for biometrics messaging

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 16 – 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 our UWS model has achieved 68.23% mAP (KPI: 70% mAP) of accuracy at 16 fps (KPI: 16 FPS), 62.5 ms. At close distances the accuracy achieved is 89.05% 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

The next steps towards the development and further enhancement of the Biometrics component are to focus on improving the accuracy and speed of the new model developed for face verification. Furthermore, low-visibility (evening and night time near street lights) images from a flying drone will be added to the collected dataset. Finally, integration and validation activities will be carried out with other components and third-party services.

In terms of integration, UWS is finalising a first prototype for integration with other components and the third-party software of domain A.

2.4 Authentication (TRU)

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 the Self-aware Data Privacy component.

2.4.1.2 Requirements

The main requirements for the authentication component are the following²⁴:

- **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.

In both cases, authentication results shall be used as input for the components of Behaviour Monitoring and Self-aware Data Privacy.

2.4.1.3 Objectives and KPIs

The MFA component contributes to the accomplishment of the following objectives and KPIs.

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 ²⁵ ; 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 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 (M30)	Current value (M20)

²⁴ Requirements enhanced since the last public deliverables according to the research

²⁵ 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

1. At least 3	1. 3
2. At least 2	2. 0
3. 2 (requirement from a demonstrator IoT solution)	3. 1

2.4.2 Technology research

2.4.2.1 Background

The MFA component is an aggregator and orchestrator of other components research results. Its overall flow is depicted in Figure 19, for the case of person authentication, and was inspired by OIDC (described in the background of the hardware-based identification and authentication, section 2.2). However, considering the project objectives it is expected that the complexity of the research done within the scope of this component will be towards the orchestration of the authentication factors that it uses, described in other sections, and not in itself.

NIST defines MFA as “a security enhancement that allows you to present 2 pieces of evidence – your credentials – when logging in to an account”²⁶. 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 are²⁷:

- 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.

To these factors, time and location can be added to strengthen the credentials verification process²⁸.

The above-mentioned 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.

Usually, for persons, a 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 authentication²⁹). To avoid the burden that may relate with a lower system usability, many systems²⁷ 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 relevant³⁰ (which is, again, person authentication). 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 research³¹. 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

²⁶ <https://www.nist.gov/blogs/cybersecurity-insights/back-basics-whats-multi-factor-authentication-and-why-should-i-care>

²⁷ <https://support.microsoft.com/en-us/topic/what-is-multifactor-authentication-e5e39437-121c-be60-d123-eda06bddf661>

²⁸ <https://www.techtarget.com/searchsecurity/definition/multifactor-authentication-MFA>

²⁹ <https://auth0.com/blog/what-is-step-up-authentication-when-to-use-it/>

³⁰ <https://blog.nordicsemi.com/getconnected/multi-factor-authentication-for-iot>

³¹ <https://link.springer.com/article/10.1007/s11227-021-04022-w>

a similar 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 process³². 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 current research findings and achievements are the following:

- A vision and architecture for the MFA process for persons and devices well-accepted between the main partners involved (TRU, UWS, ATOS and MAR)
- Definition of the first target prototype – person authentication with SSI, biometrics, and network-based credentials.
- Technical specification of all the related interfaces (see Figure 19 for interfaces understanding)
- Implementation of the first prototype, with integration testing of 2 authentication factors done and a third one ongoing (no integration in IoT solutions yet)
- Definition of integration steps with two IoT service providers (in the scope of WP5 as well)

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 first prototype of the MFA for person authentication.

2.4.3 Design specification

2.4.3.1 Logical architecture view

Figure 17 depicts a logical architecture of ARCADIAN-IoT Authentication component. Assuming the role of orchestrator, it 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.

³² <https://oauth.net/id-tokens-vs-access-tokens/>

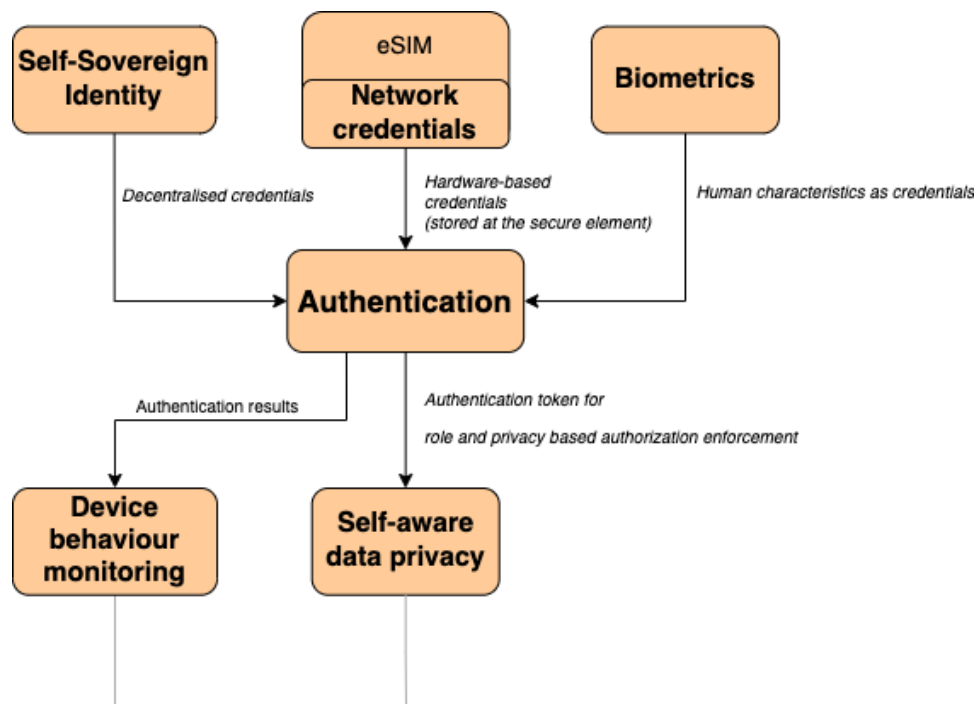


Figure 17 - ARCADIAN-IoT authentication high-level architecture

As outputs from the Authentication that feed other ARCADIAN-IoT components, there is a relation with the Behaviour Monitoring for this component to understand authentication events that may allow to infer a threat. For the Self-aware Data Privacy, after a successful authentication the Authentication issues an ID token to be used for access management control according to the user-defined privacy rules, and for role base access control (defined by the IoT service providers).

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 3 factors. The ones being considered are verifiable credentials, a hardware-based identification (eSIM/eUICC-based); and biometrics. This sub-use case is the one targeted for the first prototype as described below.

2.4.3.2.2 Device authentication

The component KPIs inform that devices should have at least robust identity mechanisms for devices. The MFA component will use for this purpose decentralized identifiers and a hardware-based identification (eSIM/eUICC-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 such a scenario is the one of Domain A, where a drone needs to identify a person that requested its services, and the identification and authentication process benefits from considering, at the same time, information from the personal device. As can be inferred, time and location will also be factors for MFA in this process.

2.4.3.3 Interface description

In terms of interfaces, the MFA interacts mainly with the IoT service provider (SP), authentication factors, Self-aware Data Privacy and Behaviour Monitoring. Details about these interfaces can be found in Table 6.

Table 6 - MFA interfaces

Sender	Receiver	Communication type	Content exchanged	Status
SP	MFA	RESTful API	Authentication request with related authentication claims	Done for P1
MFA	- SSI Authenticator - Biometrics Authenticator - Network Authenticator	- RESTful API for SSI and Network Authenticator; - RabbitMQ AMQP 0.9.1 for biometrics	Authentication claims for verification of a given requester	Done for P1 (testing with SSI missing)
- SSI Authenticator - Biometrics Authenticator - Network Authenticator	MFA	- RESTful API for SSI and Network Authenticator; - RabbitMQ AMQP 0.9.1 for biometrics	Information about claims provided validity	Done for P1 (testing with SSI missing)
MFA	SP	RESTful API	ID token (in case of authentication success)	Done for P1
Self-Aware Data Privacy	MFA	TBD	Request of ID token validity confirmation	For P2
MFA	Self-Aware Data Privacy	TBD	ID token validity confirmation	For P2
MFA	Behaviour Monitoring	TBD	Authentication results for a given ARCADIAN-IoT ID	For P2

The technical specification and sequence diagram for these interfaces was agreed among the involved partners and is shared in the common project folder (not made public to not hamper exploitation strategies definition of the involved components).

2.4.3.4 Technical solution

ARCADIAN-IoT authentication relies on multiple technologies, from multiple partners, to accomplish its objective. 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. This vision is depicted in Figure 18, which defines the well-accepted authentication flow for persons. The accepted research hypothesis for the devices' authentication flow is that it should be similar to the persons' authentication flow, but without the biometrics identification factor. Regarding the person authentication with simultaneous identification from 2 devices, this directly relates with the IoT solution of personal vigilance using drones (Domain A). In this case, partners agreed that the

main difference should be that the biometrics component, instead of using as source the biometrics data captured by the personal device, should use biometrics data captured by the drone, and should add as factors the location of each of the devices (being close enough shows that it is possible that the drone is identifying the person) and time. These research hypotheses were defined purposely to build an agnostic approach to ARCADIAN-IoT MFA, where the flow is similar in all the existent cases, varying just the number of authentication factors or sources of identification claims.

The second measure to achieve the current technical solution, after the well-accepted vision definition, was the agreement, between the involved partners on what should be the first target prototype, to be evolved iteratively after in an agile approach. The decision was to focus on prototyping the person authentication first. The rationale was that the person authentication involves all the identification factors, which allows all partners to start their research, and that the other use cases can build upon this one.

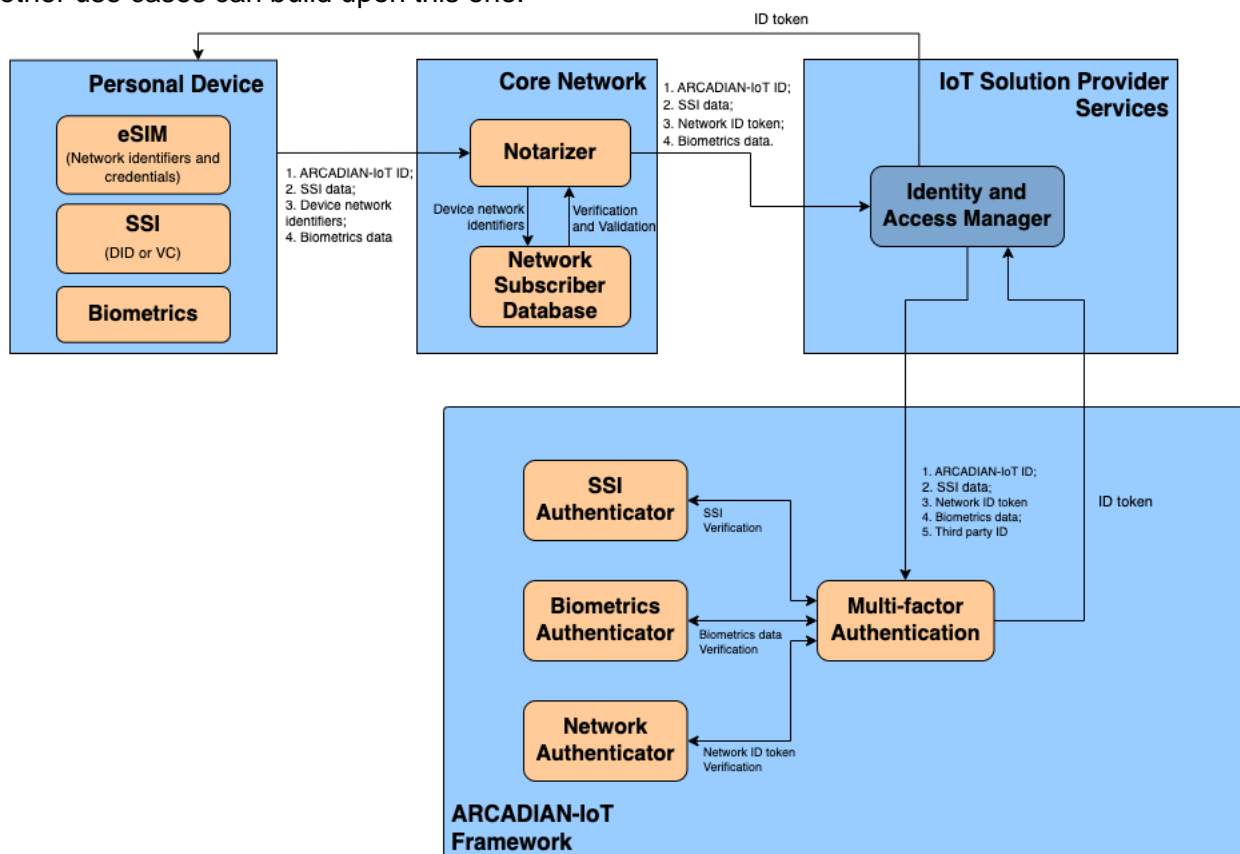


Figure 18 - Architecture from ARCADIAN-IoT MFA for persons

Thirdly, 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 in a forthcoming stage of the project. 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, for now, we are assuming 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, for the moment, 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 don't define

any access/authorization rule. For the moment, it just informs if the authentication of a given ARCADIAN-IoT ID was successful or not. In the future, if relevant, it can add more information. This strategy will be evaluated with IoT service providers within WP5 context, being aware of state-of-the-art challenges related with IoT network performance and user experience.

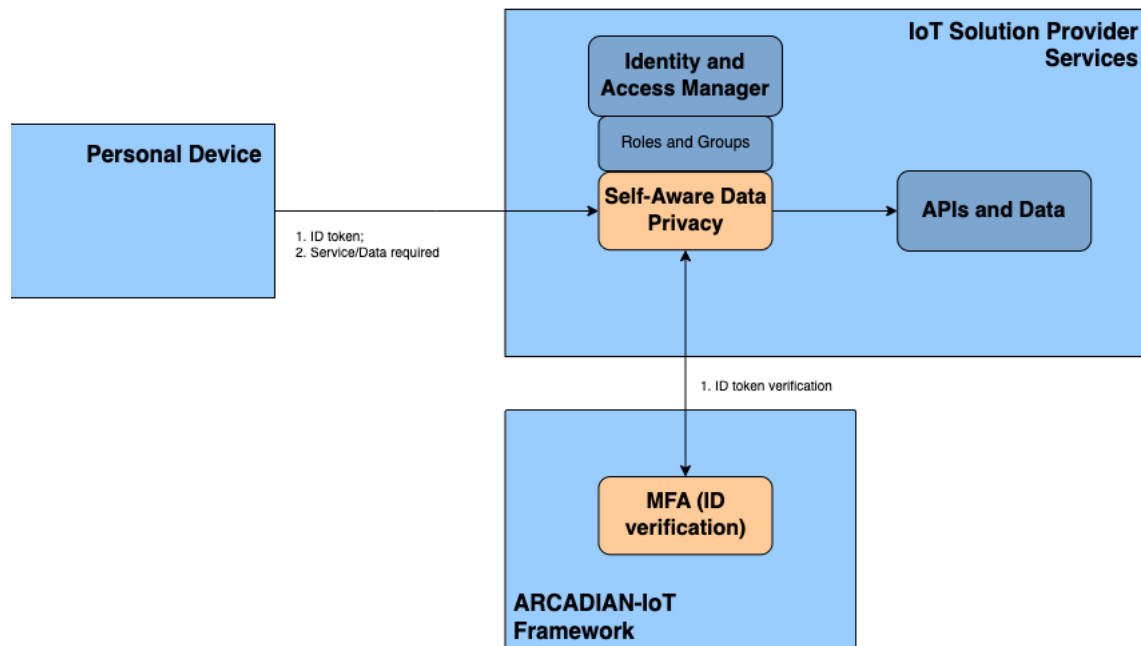


Figure 19 - Authenticated person operation flow

Lastly, the current research focused on the definition of the integration of the authentication mechanism with an authorization mechanism for the normal flow of an authenticated entity. The current research hypothesis is that after the authentication, when a person or device requests access to data or services from the service provider, it will need to present the token issued by the MFA to ARCADIAN-IoT's Self-aware Data Privacy component. This component, after confirming the token validity with the MFA, will manage the authorization rules based on privacy-related definitions or business rules specified by the IoT service providers. Figure 19 depicts the described process.

The results of the authentication of the entities (ID token) issued by the MFA component will also be used on the interaction of other components with ARCADIAN-IoT framework. An example can be the self-recovery mechanisms, directly associated with persons or devices, that need to know the requesting entity authentication results to perform its functionalities. This process will be further defined in the next research period.

2.4.4 Evaluation and results

The current MFA solution has been tested in a lab setup, particularly the integration with the biometrics and Network-based Authorization components. For the testing purpose, an emulator of service provider services was developed. While ongoing, the integration with the SSI has not been tested yet.

Testing and evaluation efforts with two domain owners (real IoT service providers) is ongoing in the scope of WP5.

2.4.5 Future work

The future work plan in what concerns the update of already prototyped components will be based on the evaluation made with IoT service providers in WP5.

Apart from those enhancements based on the evaluation to be done, in the next reporting period we expect to deliver:

- Integration with Self-aware Data Privacy component
- Integration with the Behaviour Monitoring component
- Device MFA.
- Person MFA using 2 devices as sources of data.
- Testing and evaluation of the final solution.

3 TRUST PLANE

3.1 Verifiable Credentials (ATOS)

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 [1] 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	
Benchmarking (OPTIONAL)	
Not Applicable	
Target value (M30)	Current value (M20)
1	0

KPI scope	
Interoperability with at least one eIDAS identity schema.	
Measurable Indicator	
Issue person Verifiable credential with an eIDAS compatible schema.	
Benchmarking (OPTIONAL)	
Not Applicable	
Target value (M30)	Current value (M20)
1	0

KPI scope	
Enable, at least 3 multiple simultaneous identification approaches for persons.	
Measurable Indicator	
Support Verifiable Credential identification from a person´s mobile wallet.	
Benchmarking (OPTIONAL)	
Not Applicable	
Target value (M30)	Current value (M20)
1	0

KPI scope	
Support, at least two robust identity mechanisms for devices and apps/services.	
Measurable Indicator	
Devices and apps/services support Verifiable Credentials at least in one domain.	
Benchmarking (OPTIONAL)	
Not Applicable	
Target value (M30)	Current value (M20)
2	0

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

ATOS is in the process of building a Self-Sovereign Identity solution called Ledger uSelf with the aim to simplify the adoption and integration of Self-Sovereign Identity by Service Providers. The existing prototype asset provides 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. Today, the Ledger uSelf solution has 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.

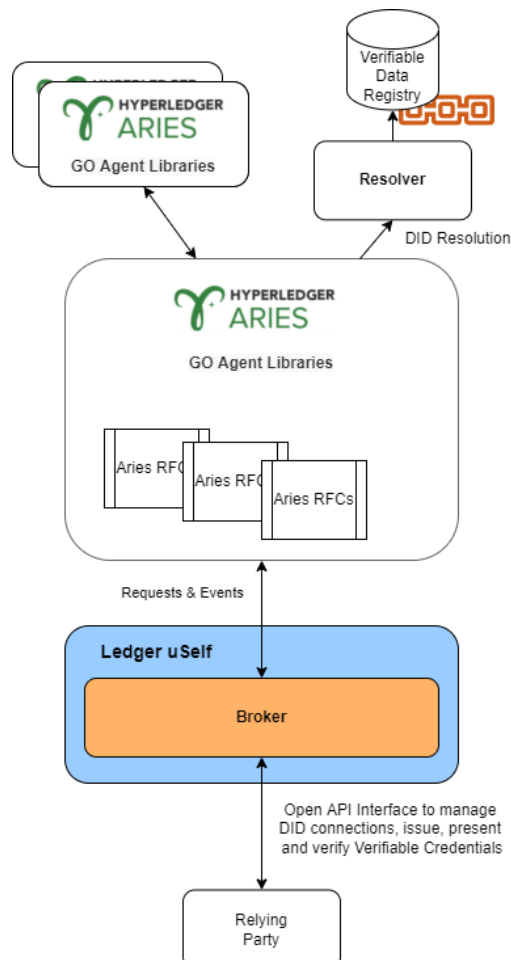


Figure 20 - Ledger uSelf built on top of Hyperledger Aries GO Agent

The current Ledger uSelf Broker prototype is implemented in Kotlin and is a completely separate component from the Aries GO Agent.

The following sub-sections investigate the current state-of-the-art in this area 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. Here we examine some of the solutions available today:

- **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

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

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.

- **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 21 - 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 22 - 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 here³⁴.

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.

³⁴ <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 here³⁵. 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 2.1.2.1.4).

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.

³⁵ <https://github.com/hyperledger/aries-framework-go/releases>

3.1.2.3 Produced resources

In addition to enhancing the ledger uSelf SSI solution as described in the following section 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 IoT Device's SSI Agent*

An IoT Device is able to 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.

3.1.3.1.3 *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.

3.1.3.1.4 *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.

3.1.3.1.5 *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.

3.1.3.1.6 *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.

3.1.3.1.7 *Verify a constrained 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 proving that it is in possession of the DIDs private key.

3.1.3.1.8 *Service Provider service registers 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".

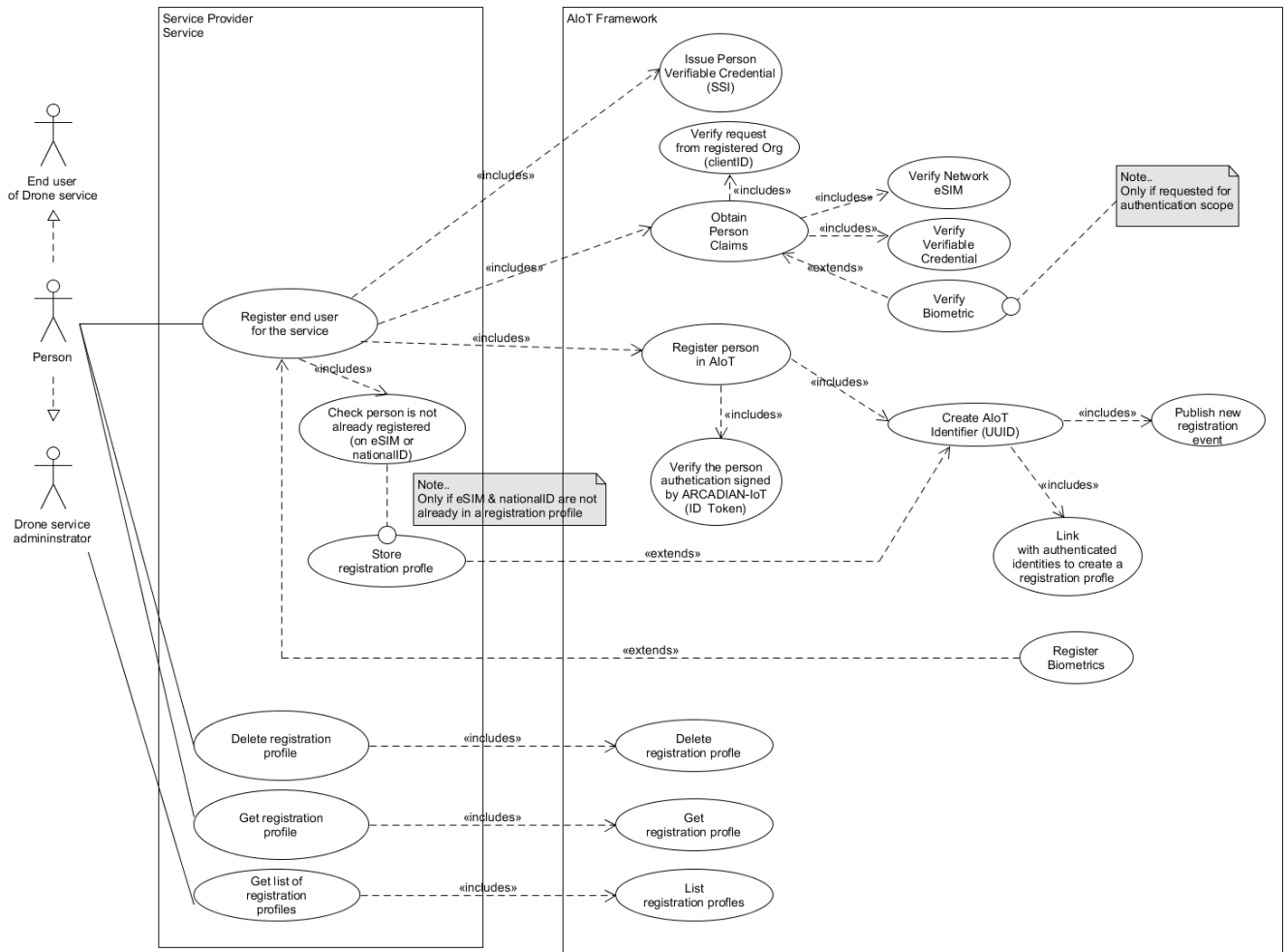


Figure 23 - Register Person in ARCADIAN-IoT Framework by a SP service

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).

3.1.3.1.9 Service Provider service registers an IoT-Device in the ARCADIAN-IoT framework

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

3.1.3.1.10 Service Provider service updates a registered Person or IoT-Device identity in the ARCADIAN-IoT framework

Previously onboarded Persons and IoT devices are updated by a Service Provider service 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).

3.1.3.1.11 Service Provider service deletes a registered person or IoT-Device in the ARCADIAN-IoT framework

A Service Provider service 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.

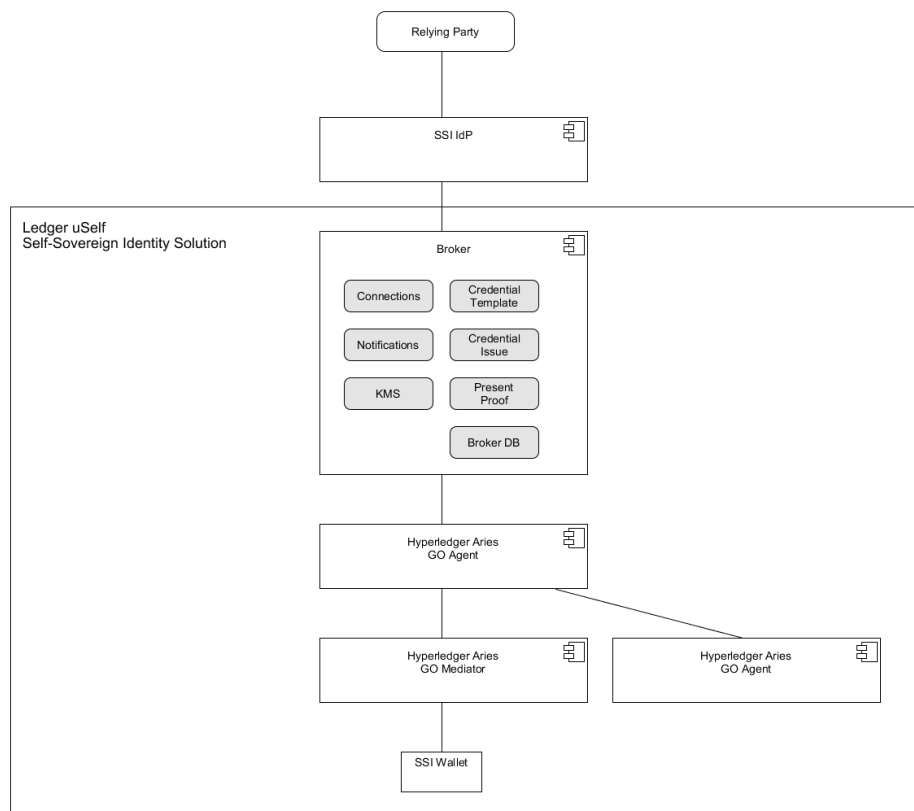


Figure 24 - Ledger uSelf Broker Self-Sovereign Identity Solution + SSI IdP

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)

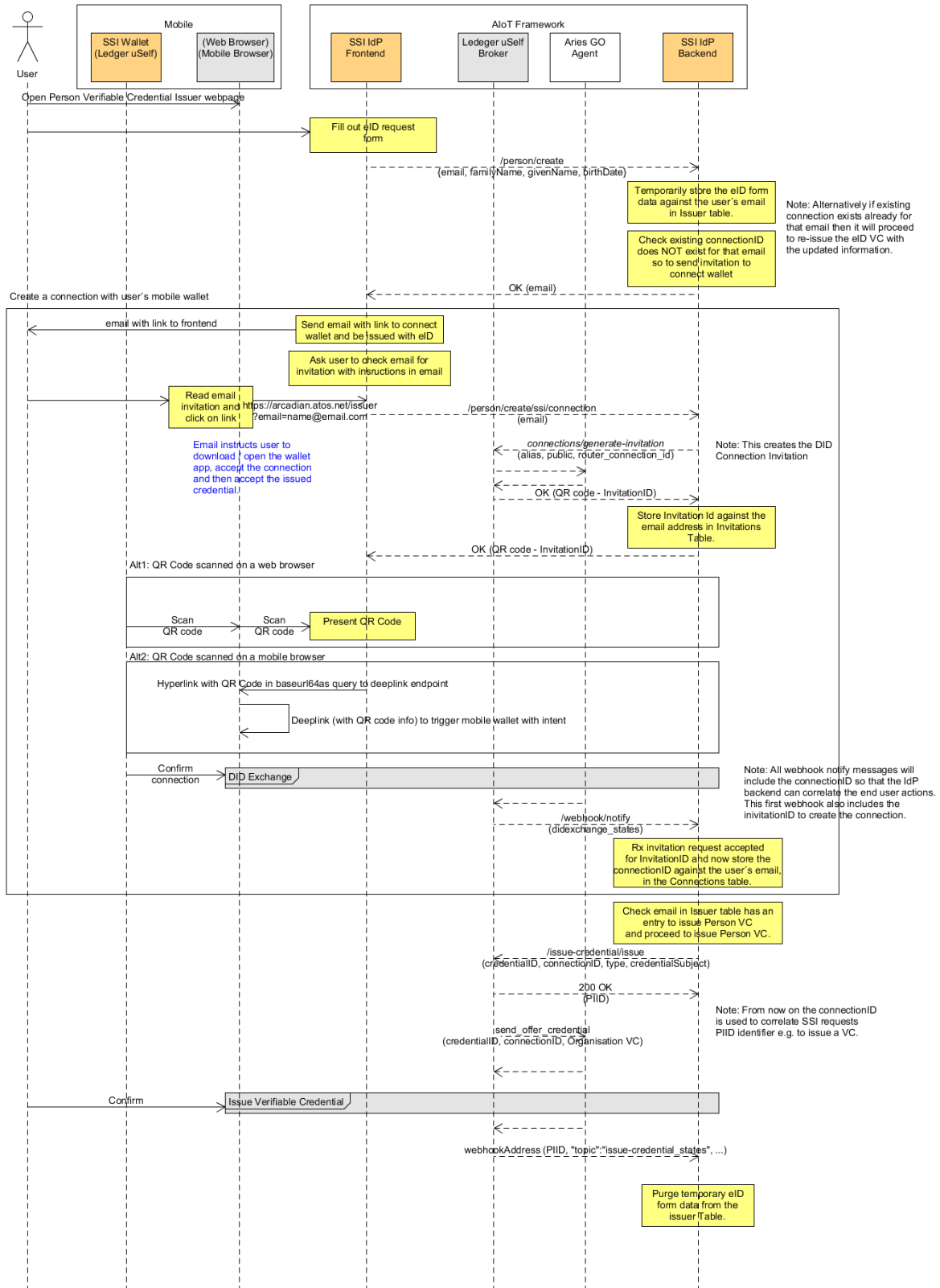


Figure 25 - Issue a Person VC



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

To be confirmed for final prototype.

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

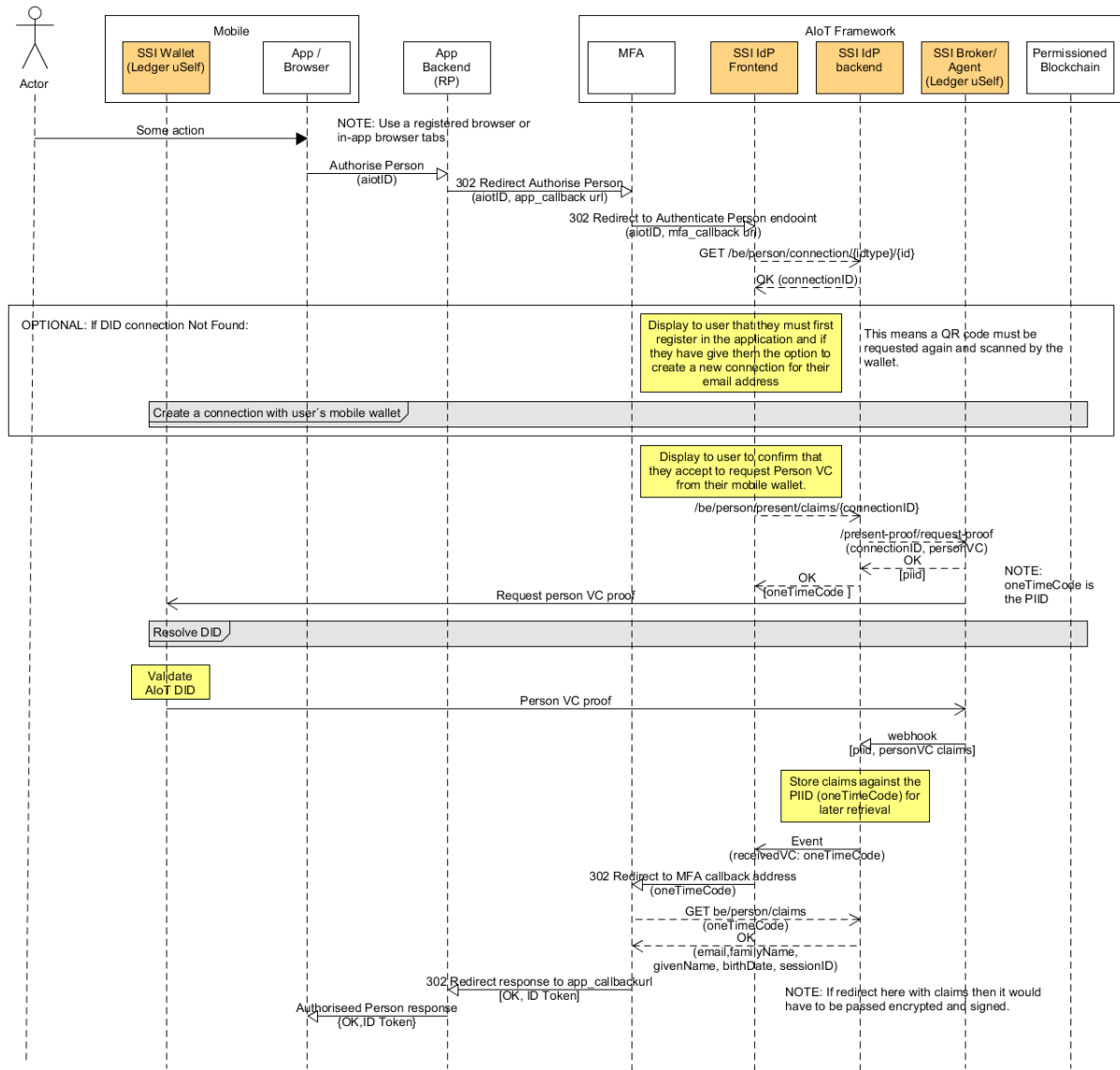


Figure 26 - Verify a Person VC and provide identity claims

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.

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

To be confirmed for final prototype.

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

See section 3.1.3.1.3.



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

See section 3.1.3.3.4.

3.1.3.3.7 Verify a constrained IoT Device's SSI Agent with its PUBLIC DID (P2)

To be confirmed for final prototype.

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

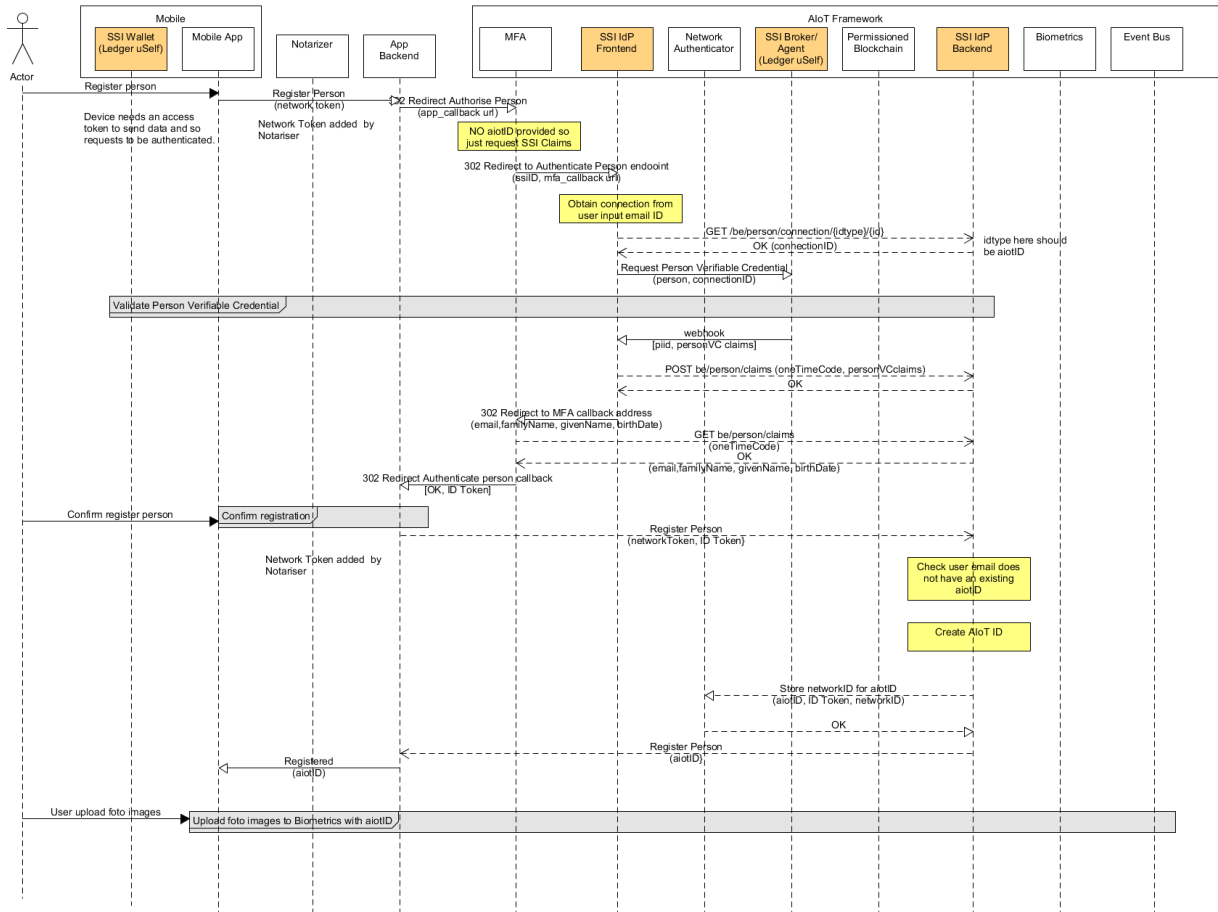


Figure 27 - Service Provider service registers a Person

3.1.3.3.9 Service Provider service registers IoT-Device in the ARCADIAN-IoT framework (P2)

To be confirmed for final prototype.

3.1.3.3.10 Service Provider service updates a registered Person identity in the ARCADIAN-IoT framework

This is similar to the initial register flow from the viewpoint of the Service Provider but uses a PUT instead of a POST.

3.1.3.3.11 Service Provider service updates a registered IoT-Device identity in the ARCADIAN-IoT framework

To be confirmed for final prototype.



3.1.3.3.12 Service Provider service deletes registered person from the ARCADIAN-IoT framework (P1)

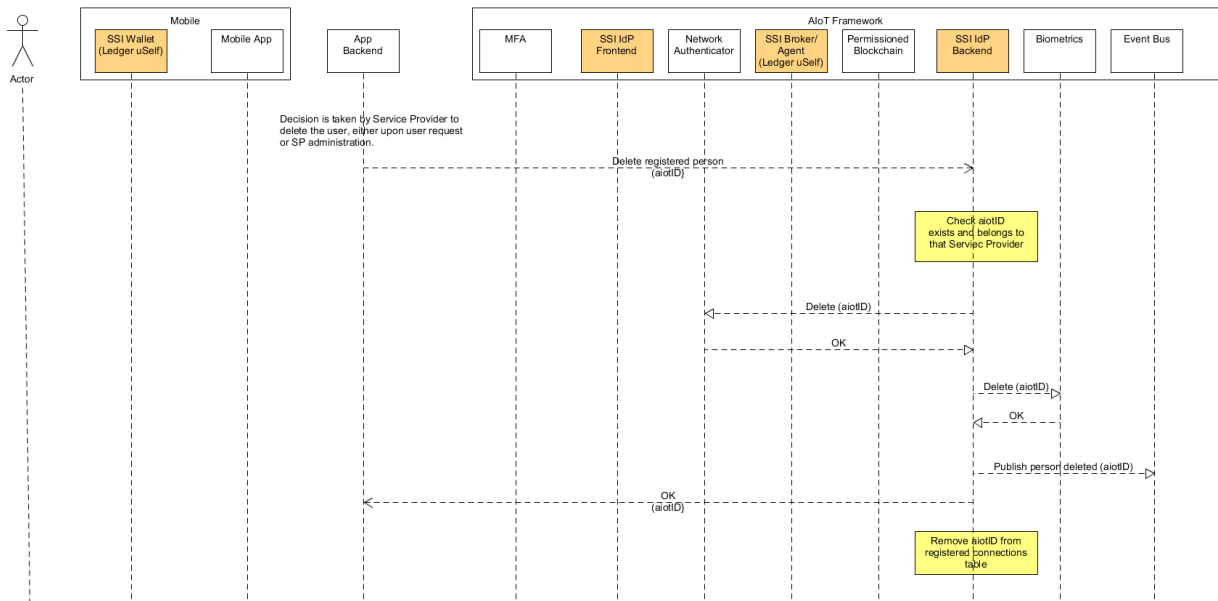


Figure 28 - Service Provider deletes a registered Person that it previously registered

3.1.3.3.13 Service Provider service deletes a registered IoT-Device from the ARCADIAN-IoT framework (P2)

To be confirmed for final prototype.

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.

ARCADIAN IOT SSI module API Specification 0.0.5 OAS3

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

Servers

/

Authorize

SSI IdP Backend 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. v
- POST /be/person/create/ssi/connection/{email} BE Request to create a DID connection invitation with the user's wallet. v
- 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). v
- POST /be/person/claims/{oneTimeCode} BE Request to get authenticated Person claims. v
- GET /be/person/claims/{oneTimeCode} BE Request to get authenticated Person claims. v
- POST /be/person/register Register a person in AIOT Framework. v
- PUT /be/person/register Update to a registered person in AIOT Framework. v
- POST /be/service/register Register an SP service in AIOT Framework. v
- 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. v
- 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. v

SSI IdP Frontend Person Identity Management Frontend ^

- POST /person/authenticate FE Request to authenticate a person. v

Figure 29 - SSI IdP Interface description



3.1.3.5 Technical solution

3.1.3.5.1 Deployment architecture view (optional)

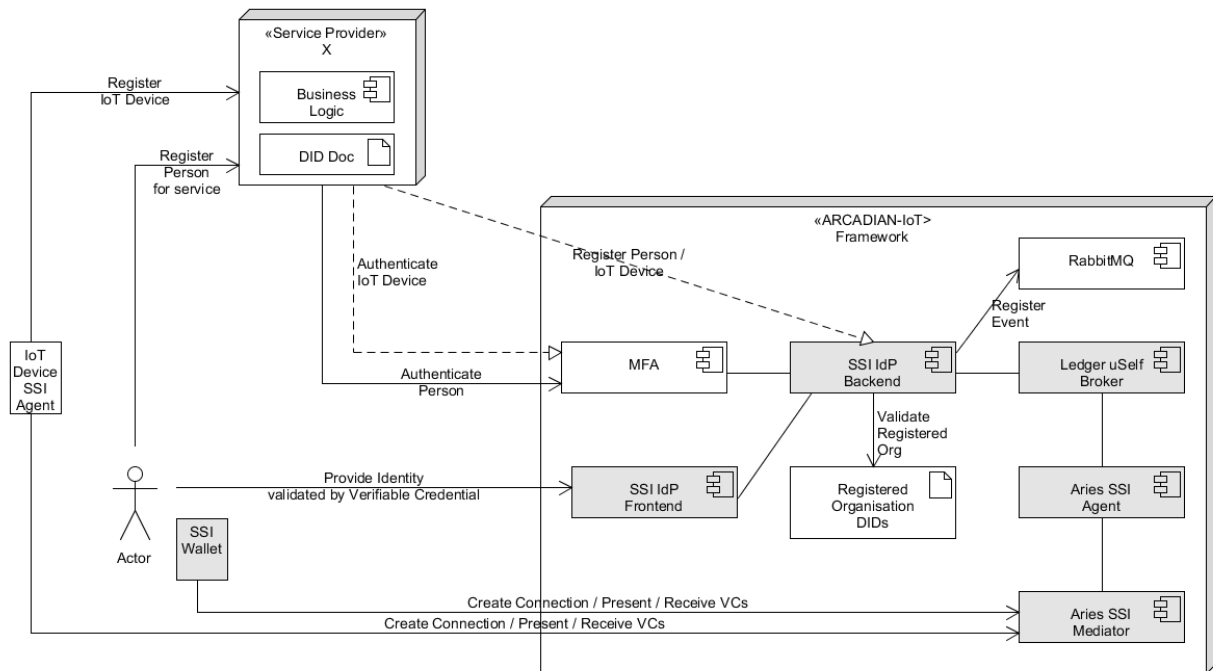


Figure 30 - Self-Sovereign Identity deployment in the ARCADIAN-IoT Framework

3.1.3.5.2 Domain model

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

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

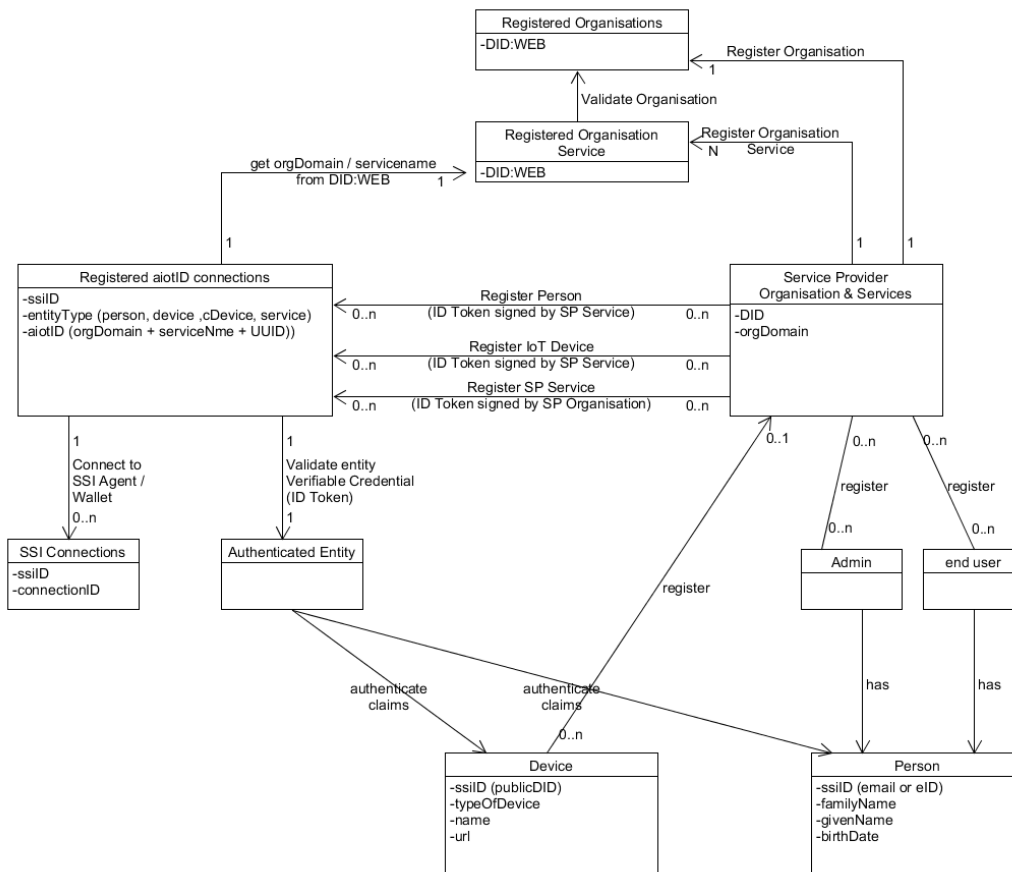


Figure 31 - SSI IdP Registered Entities

3.1.3.5.3 API specification

The Open API specification for the SSI IdP is uploaded to ARCADIAN-IoT GitLab project here: https://gitlab.com/arcadian_iot/verifiable-credentials/-/blob/main/ssildP/interface/ssildPopenapi.yml

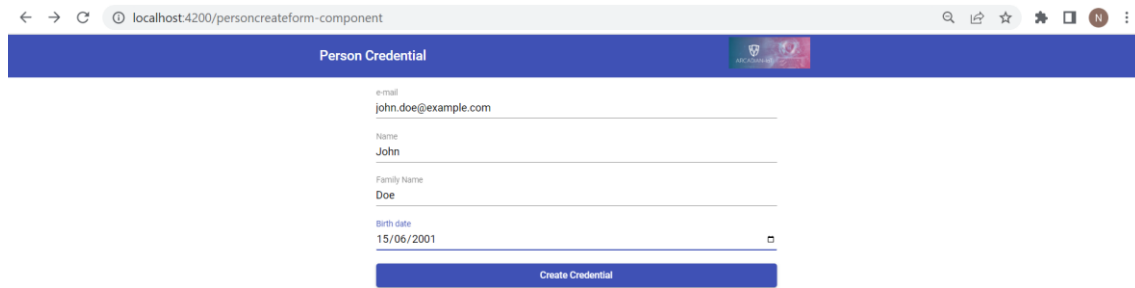
3.1.3.5.4 Frontend design

SSI IdP Issuer

The frontend supports the issuing of Person’s 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.





The screenshot shows a web browser window with the URL `localhost:4200/personcreateform-component`. The page title is "Person Credential". The form contains the following fields:

- email: john.doe@example.com
- Name: John
- Family Name: Doe
- Birth date: 15/06/2001

At the bottom of the form is a blue button labeled "Create Credential".

Figure 32 - 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.



Figure 33 - 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 the email address in first prototype P1. In the final prototype P2 it may be additionally supported a national identity.

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.

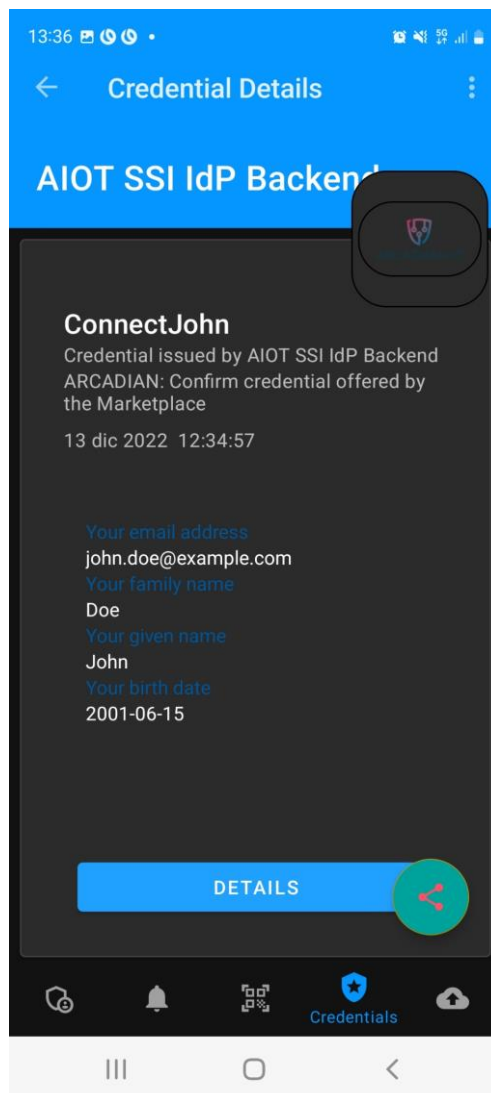


Figure 34 - Ledger uSelf mobile SSI Wallet UI

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. In the first instance this will be issued by the ARCADIAN-IoT SSI Agent for the first prototype P1 and later it is under consideration in the final prototype P2 to be issued with a national eID outside of the framework's SSI Agent.

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",
    "https://json-ld.org/context/person.jsonld"
  ],
  "id": "http://issuer.arcadianiot.eu/8a329249-d437-4c42-9d68-f6450215a11f",
  "type": ["VerifiableCredential", "PersonCredential"],
  "issuer": "did:web:issuer.arcaiaianiot.eu",
  "issuanceDate": "2022-01-01T19:23:24Z",
  "expirationDate": "2032-01-01T19:23:24Z",
}
```

```

"credentialSubject": {
  "email": "ross.little@ATOS.net",
  "familyName": "Little",
  "givenName": "Ross",
  "birthDate": "2001-04-18"
},
"proof": {
  "type": "Ed25519Signature2018",
  "created": "2021-11-13T18:19:39Z",
  "verificationMethod": "did:web:issuer.arcdaianiot.eu:#key-1",
  "proofPurpose": "assertionMethod",
  "jws":
"z58DAdFfa9SkqZMVPxAQpic7ndjjhghjhjSayn..1PzZs6ZjWp1CktyGesjuTSnmnm76mmwRd
o
      WhAfGFCF5bppETSTojQCrfFPP2oumHKtz"
}
}

```

Person 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 P1 prototype is the user's email address.

Additionally, the SSI IdP will maintain a table of all registered users with the ARCADIAN-IoT ID, the SSI ID and the entity type (in this case person).

Therefore, Person authentication with the user's 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 VC from the user's SSI wallet, based on the user's SSI ID. In this case the backend will lookup the connection persisted for that SSI ID to then request the Person VC to the SSI Wallet.
- 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 VC for the associated connection.

SSI Webhooks

Communications with the user's mobile SSI Wallet 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:

- when the user has accepted a connection
- when the user has accepted an issued Verifiable Credential
- when the user has presented 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 service. ARCADIAN-IoT identity is managed within the framework as follows:

- A specific Service Provider service is responsible for registering their users to the ARCADIAN-IoT framework.
- The same end users can be registered by other Service Provider services to ARCADIAN-IoT, each with a different aiotID, so the end user can have multiple aiotIDs: one per Service Provider service.
- When an SP wants to delete its user and all its associated data it only removes the ARCADIAN-IoT component service 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 service 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:serviceName:UUIDv4"

```

The "orgDomain" and "serviceName" is obtained from the registered DID WEBS of the Service Provider (see section 2.1.3.2.2) and the UUID v4 is generated upon registration.

A non-normative example is:

```

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

```

IoT Device Registration

To be confirmed for final prototype.

Service Provider Registration

A JSON file is configured in the SSI IdP backend to provide a table of all registered Service Provider organisation Decentralized Identifiers.

A non-normative example is given as follows:

```

{
  "organizationDIDs":[
    "did:web:ATOS.com",

```

```

"did:web:ipn.pt",
"did:web:truephone.com",
"did:web:xlabs.si"
]
}

```

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 ssiID 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.

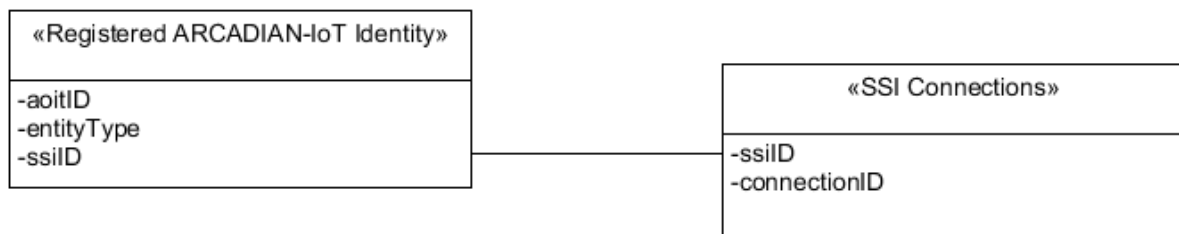


Figure 35 - SSI IdP Data Model

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
- ssiID
- 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 IoT-Devices the Broker is being re-written in GO so to be able to integrate it with the Hyperledger Aries GO Agent in one package. This is planned for P2 and will provide a leaner implementation for more efficient operation in devices with reduced resources, as compared with the typical cloud resources.

3.1.3.5.8 Security aspects

The framework security between the components is not added in P1 and will be added once the interwork is proven in P2. Mutual TLS is currently the candidate technology to provide trusted and secure communications between the ARCADIAN-IoT framework components.

3.1.4 Evaluation and results

The SSI IdP implementation is currently nearing completion of development and test in a local environment and is under integration with the current version of ledger uSelf for supporting Person VCs. Implementation of integrated broker and SSI Agent in GO is ongoing.

3.1.5 Future work

The following items are under the scope of future work:

- Complete integration for authentication and integration flows.
- Agree final interwork for IoT Devices, VC issuing, registration & authentication
- Replace current Ledger uSelf SSI Agent and Broker separate components with the one integrated SSI Agent / Broker component
- Implement & integrate IoT Devices with SSI IdP & new integrated SSI Agent Broker
- Investigate to add support for SIOP OIDC4VP protocols as identified in section 3.1.2
- Agree support for constrained MCU IoT-Devices in Domain B considering SIOP / DPop³⁶ protocols and use of eSIM for public / private key needed to support Decentralized identifiers.
- Investigate to support eIDAS Bridge / EBSI - ESSIF interoperability for supporting trusted national eIDs
- Support BBS+ signatures to Present VC from the SSI Wallet

³⁶ <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-dpop-02>

3.2 Authorization: Network-based Authorization enforcement and authorization distribution (TRU)

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, the authorization information distribution to devices, both based in entities (e.g. devices) trustworthiness information or security reputation, and the self-aware data privacy, where users (or IoT Service Providers) define authorization rules to allow access to their data. In this section we will focus on the component that, on one hand, enforces trust-based authorization rules in the cellular network core (between devices and internet services), and on the other, informs devices' secure element (eSIM, in the case) of the trustworthiness level of the device where it is at.

3.2.1.2 Requirements

The requirements for this component are the following³⁷:

- **To provide a dynamic Network-based Authorization enforcement based on entities trustworthiness level and security policies:** A network-based enforcement tool (placed in the core network) to control devices, persons, and services communication/interaction based on those entities' 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 or 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 (M30)	Current value (M20)
1. Dynamic bidirectional communication enforcement according to security policies and trustworthiness level 2. Near real time	1. Authorization enforcement just in the direction of the subscribers / devices to the internet, according to entities trustworthiness level. 2. Near real time

³⁷ Requirements enhanced since the last public deliverables according to the research

KPI scope	
This component also contributes to the objective of <i>self and coordinated healing with reduced human intervention</i> , by informing the eSIM of device's trustworthiness information, triggering the subsequent eSIM-based protection and recovery actions. In this case the research focuses on informing the eSIM of devices trustworthiness level for it to be able to act as an enabler of self-protection and self-recovery.	
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. Time to implement self-protection or self-recovery actions after the trustworthiness information arrive the eSIM 4. Number of different devices where the innovation is demonstrated	
Target value (M30)	Current value (M20)
1. Y	1. Y
2. Y	2. Y (preliminary prototype)
3. < 2 seconds	3. ~1 second (with 40% of network-related outliers ³⁸ between 9 and 24 seconds)
4. At least 2	4. 1

3.2.2 Technology research

To accomplish the research objectives of this component the work had the following phases:

- (1) define a unified vision with the partner responsible for ARCADIAN-IoT reputation system, which will be directly integrated with this component, being therefore critical to the action of the Network-based Authorization.
- (2) research on how to create a network testbed able of accommodating the Network-based Authorization component – the network testbed allows testing the technology with virtual devices (for research purposes, throughout the period of the project). At the end of the project, for technology demonstration, the network testbed will connect real devices with real networks.
- (3) have a first working prototype of the component with basic security rules being enforced automatically. The final prototype will have the security rules considered needed, which are possible to implement in a core network PCF.
- (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 authorization rules. The simplest high-level example of this process is the one of a device that, because it has a high reputation has access to all the systems and data it requests, while one with low trust reputation has communication restrictions, for example, can only communicate with ARCADIAN-IoT recovery services.

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

³⁸ The events designated here as *network-related outliers* may not be controllable due to potentially being related with the radio access network hardware, and therefore vary according to location and service providers (and other factors uncontrollable by TRU)

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 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 beyond ARCADIAN-IoT Self-Recovery. The same component also triggers information to the devices' secure element (to ARCADIAN-IoT applet in the eSIM 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) architecture³⁹ provides access, resource, and quality of service (QoS) control⁴⁰ to mobile networks. Two components of this architecture are the Policy and Charging Rules Function (PCRF) and the Policy and Charging Enforcement Function (PCEF), or the Policy Control Functions (PCF), for 5G, an evolution of the PCRF/PCEF with similar functionalities but adapted to 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, 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

³⁹ <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=810>

⁴⁰ <https://www.netmanias.com/en/post/techdocs/10997/lte-pcrf/policy-and-charging-rules-function-pcrf-in-lte-epc-core-network-technology>

that require a dynamic PCC⁴¹. 3GPP's PCC architecture describes an AP 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 AP extracts session information and media-related information from the application signalling and provides application session-related information to the PCRF using the Rx⁴² 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 messages⁴².

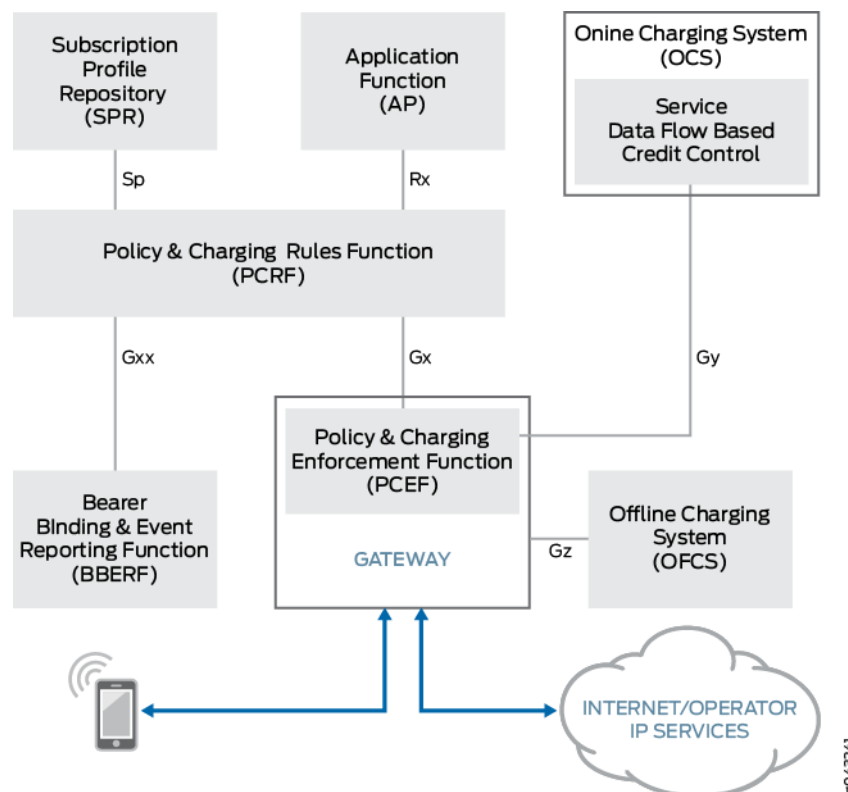


Figure 36 - 3GPP's PCC Architecture overview⁴¹

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 PCRF⁴³.

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 PCEF⁴⁴ and can be activated, modified, and deactivated in runtime.

In the context of ARCADIAN-IoT, PCRF/PCEF solutions can be leveraged to efficiently and dynamically route and prioritize network traffic⁴⁰ as a means of providing trust-based authorization

⁴¹ <https://www.juniper.net/documentation/us/en/software/junos/subscriber-mgmt-sessions/topics/topic-map/3gpp-policy-charging-control-provisioning-accounting.html>

⁴² <https://www.netmanias.com/en/post/techdocs/10997/lte-pcrf/policy-and-charging-rules-function-pcrf-in-lte-epc-core-network-technology>

⁴³ https://www.3gpp.org/more/1629-rx_interface

⁴⁴ <https://www.netmanias.com/en/?m=view&id=techdocs&no=11863>

inside the network. PCRF/PCEF use policy-based authorization, however, as seen in ⁴⁵, 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, and since we expect to use the current PCC architecture, 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. Open5GS “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.”, i.e., 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 network testbed choice.

Figure 37 shows Open5GS architecture, which depicts its capabilities as network testbed. 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).

⁴⁵ <http://reverse.net/publications/download/REVERSE-RP-2005-116.pdf>

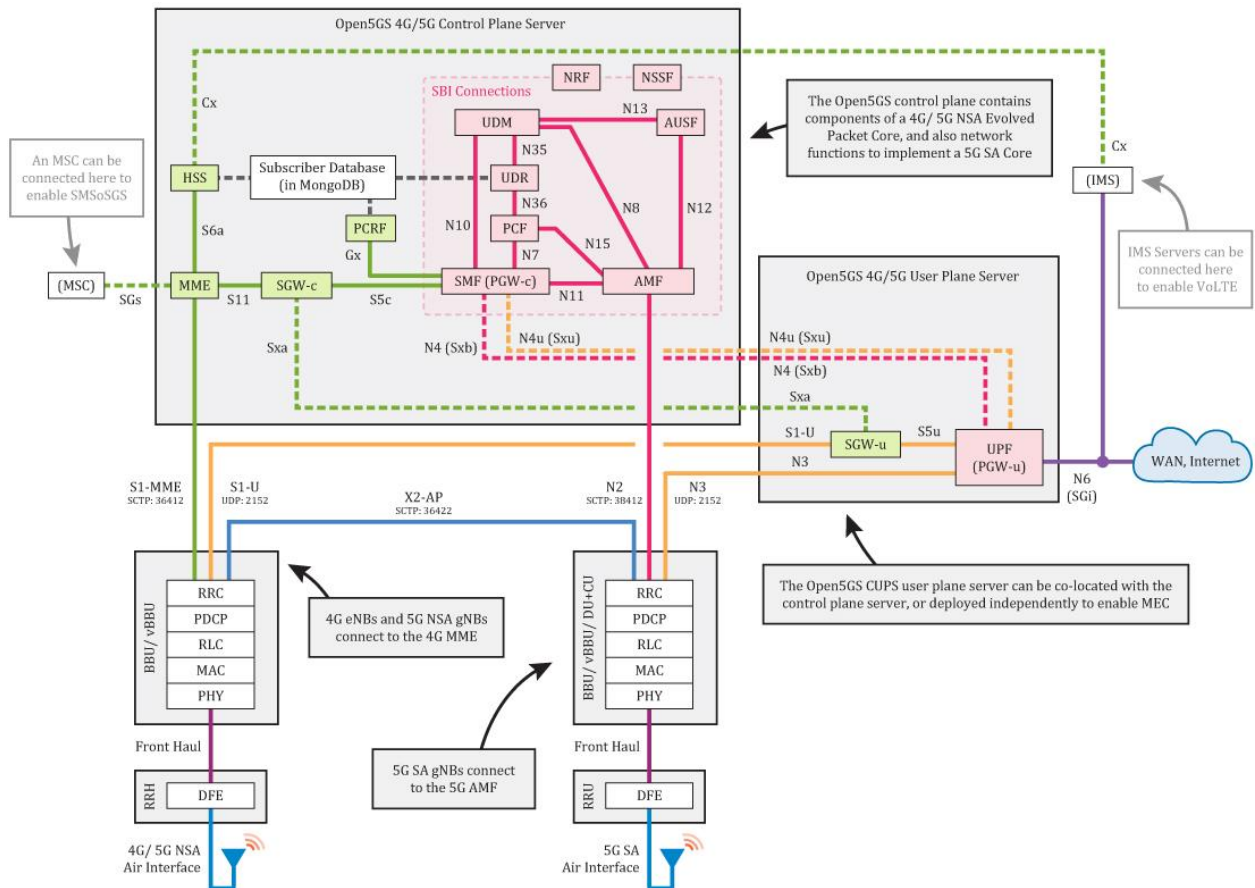


Figure 37 - Open5GS architecture⁴⁶

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

- Website⁴⁷ with blog, tutorials, and other documentation.
- GitHub repository⁴⁸ 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 vision over the features of the Network-based Authorization component is stable and well-accepted between the involved partners (the one responsible for the component and the ones with interfacing technologies). The hypothesis of using PCRF/PCEF (4G) or PCF (5G) for trust-based policy enforcement in the network core, seems quite robust. These technologies are successfully proven in highly scalable mobile scenarios for real time policy enforcement. Therefore, no reason is foreseen for not being possible to apply it for enforcing security related communication policies in the envisioned IoT scenarios. The research focus is, therefore, in the novel component that allows the integration between the Reputation System, the PCRF/PCEF or

⁴⁶ <https://open5gs.org/open5gs/docs/guide/01-quickstart/>

⁴⁷ <https://open5gs.org/>

⁴⁸ <https://github.com/open5gs/open5gs>

PCF present in the network testbed, and the standardized network OTA services, to distribute the trust information to the devices secure element.

To start validating the hypotheses formulated, the first prototype had the following characteristics:

1. Network testbed: Open5gs
2. One to two virtual devices in the network
3. Network-based Authorization system prototype considering just boolean reputation scores and simple trust-based policies:
 - a. Reputation of 0 means a compromised device that can't access internet services until its reputation is recovered to 1;
 - b. Reputation of 1 means a trustworthy device that can communicate with any service in the internet.
4. Trust information distribution to devices secure element (eSIM) via OTA services.
5. Deployment of the network testbed, including the virtual devices, and of the authorization component in AWS for its integration with ARCADIAN-IoT reputation system.

The research process allowed to, not only functionally validate the aspects related with the hypothesis raised for the solution, but also to understand limitations of the approach, and define next steps. Some of the research findings and achievements can be found below:

- **Functional validation:** The current setup works as expected. In one hand, the security policies are applied to devices according to their trustworthiness level (just 0 or 1 in this first prototype) in near real time. In the other, the trust information is sent by the authorization component via GSM OTA services to the eSIM secure element, and the information is received by ARCADIAN-IoT security applet in two quite different devices: an Android smartphone and a Linux-based IoT device (however, at the moment, self-protection and self-recovery only works in the IoT device; the device middleware for communication with the security applet is not implemented for Android devices yet).
- **Real time authorization enforcement:** The real time authorization enforcement was a challenge found during the research using Open5GS as testbed. While the rules that already exist on Open5GS PCRF were applied in real time, the newly generated rules, by default, weren't immediately enforced. This caused that a new rule was just applied when the related device was reset. A new Diameter Agent, able of informing the network components of the new rule was the solution to overcome this challenge.

3.2.2.3 Produced resources

Table 7 depicts the status of the current Network-based Authorization component, including the network testbed.

Table 7 - Network-based Authorization component current produced resources

#	Subcomponent	Brief description ⁴⁹	Prototyping status
1	Network testbed	Based on Open5GS and deployed in AWS, it allows to integrate the Network-based Authorization component for research purposes, including for the demonstration scenarios integrating real devices in real networks.	The current prototype is ready for receiving reputation-based policies for binary communication enforcement (allow all communication for trustworthy devices; block all

⁴⁹ The brief description is for the whole component as expected according to the requirements. The prototype status just describes the current prototype.

			Includes the needed network core elements, and the additional diameter agent for real time enforcement of the security policies.	communication to compromised devices). This prototype also informs the eSIM security applet of the subscriber devices regarding the authorization information, according to the same policies, allowing it to act accordingly. In a first stage, a local deployment (in a researcher laptop) of the setup was done and tested successfully. The current version was deployed in AWS for integration with ARCADIAN-IoT Reputation System and successfully tested with the project messaging (pub/sub) infrastructure. A demo of a working prototype has been done in an internal (TRU) workshop, to the partners in a consortium meeting, and to the European Commission reviewers of Y1 deliverables.
2	Network-based Authorization – reputation interpreter and network security policy generator		<p>Receives trust-based policies - which rules should be applied according to entities reputation scores. Receives trust-based reputation scores from the entities in ARCADIAN-IoT ecosystems (persons, devices, services). Performs the conversion of the ARCADIAN-IoT ID, which comes from the Reputation System attached to the reputation score, to the network subscriber ID (e.g. IMSI); Automatically generates PCRF rules according to the trust-based policies and each entity reputation score;</p> <p>Upon a new policy related to a reputation change, requests the diameter agent to inform Open5GS network components of the new policy to be enforced.</p> <p>And triggers an OTA to ARCADIAN-IoT eSIM security applet to perform device self-protection or self-recovery (depending on the reputation change).</p>	

A relevant result so far is that the hypothesis of using PCRF or PCF to enforce security rules according to the needs of ARCADIAN-IoT seems to be proven. This assumption will prove to be false if the research shows not to be possible to enforce rules with the granularity found to be necessary, which is not foreseen. For the moment, according to the security policies provided by the Reputation System is just expected the need for bidirectional control of the communication according to entities trustworthiness. Specifically, the allow or block communications is considered relevant; as well as to define a set of specific allowed or denied domains for a given entity (e.g., a given device with a given reputation score can only communicate to a given Internet service or receive commands from a given Internet service).

The major challenge found so far related with in the research and development of the network testbed environment, particularly with having it enforcing security rules in near real time. This challenge was overcome with a new subcomponent for Open5GS for propagating diameter messages for the network components to trigger the new rule.

3.2.3 Design specification

3.2.3.1 Logical architecture view

Figure 38 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 automatically translate it in knowledge understandable by the cellular networks' functions that can act accordingly. By using PCFs or PCRFs/PCEF, mechanisms known for their scalability and performance, programmatically orchestrated with the security-based Reputation System, we aim to automatically act 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.

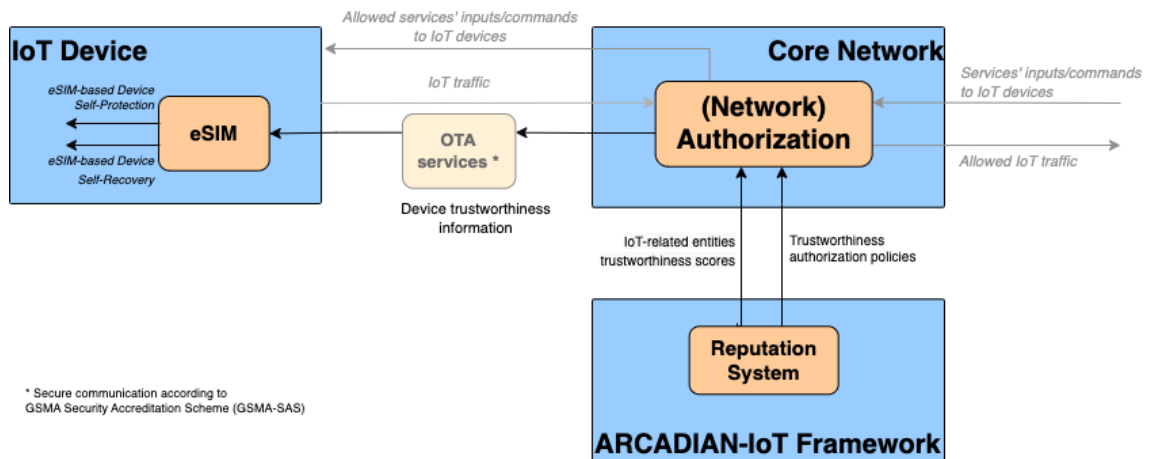


Figure 38 - ARCADIAN-IoT Network-based Authorization high-level architecture

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 deliverable D3.2, in the Hardened Encryption section.

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 eSIM security applet (further described in the context of the Hardened Encryption component). The interfaces are the following (the Network-based Authorization component should be seen as a consumer of the content exchanged):

Table 8 - Network-based Authorization interface status

Component	Communication type	Content exchanged	Status
-----------	--------------------	-------------------	--------

Reputation system (publisher)	RabbitMQ AMQP 0.9.1	Entities reputation scores	Done from TRU side for P1
Reputation system (publisher)	RabbitMQ AMQP 0.9.1	Reputation-related policies to be enforced	Done from TRU side for P1
eSIM	GSM OTA services	Trust/reputation information	Done

3.2.3.3 Technical solution

To depict the current technical solution, the architecture shown in Figure 39, extends the previously analysed Figure 38, focusing now the technical details of the Network-based Authorization component.

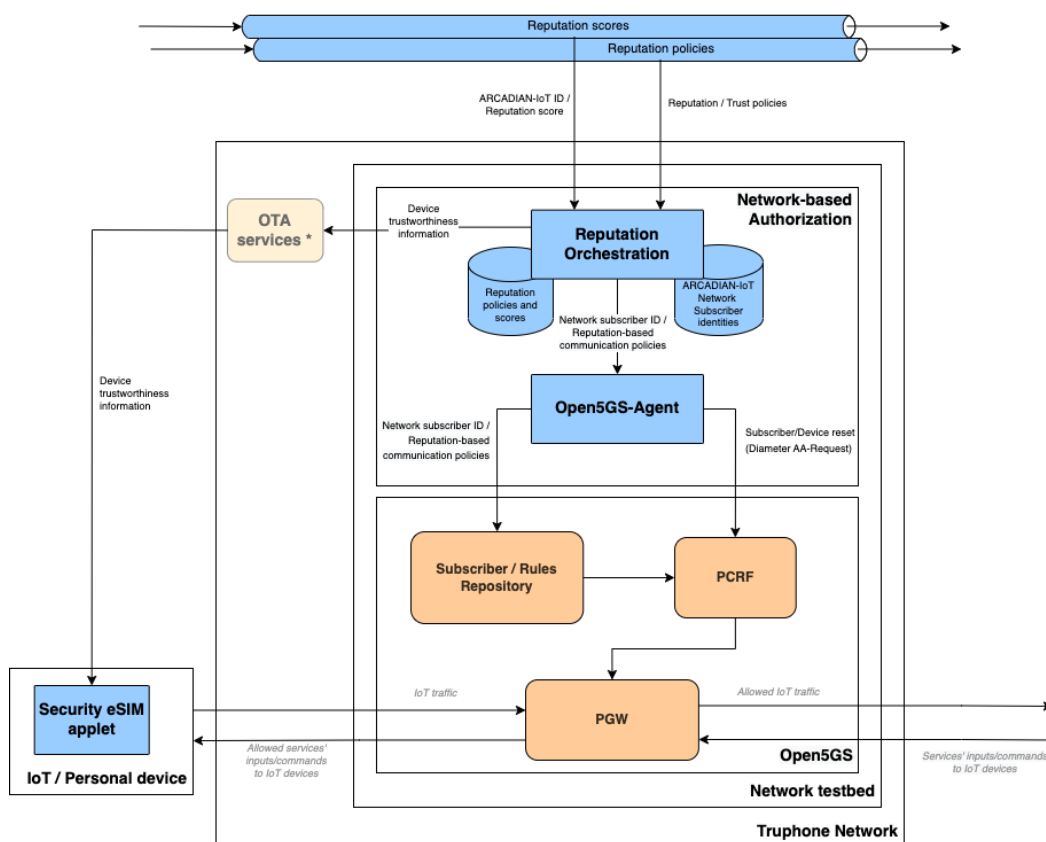


Figure 39 - Network-based Authorization current technical architecture

The integration vision defined by the involved parties (TRU for the Network-based Authorization, and UC for the Reputation System) assumed that the communication between the Reputation System and Network-based Authorization components will be made in a publish-subscribe approach, with the Reputation System publishing information to two different topics, one with the trust-related policies (i.e., which communication rules should be applied to devices/people/services according to their reputation score) and another with the reputation score for a given identifier. The trust-related policies are expected to be stable, with no frequent changes happening. Each entity reputation score is expected to be more volatile, subject to security or privacy related events detected and managed within ARCADIAN-IoT’s ecosystem.

As can be seen in Figure 39, both the Open5GS (open-source implementation of 4G / 5G core network functions) and ARCADIAN-IoT Network-based Authorization component exist in a network testbed, deployed in AWS for the moment.

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

- a. To receive and store reputation scores and policies
- b. To consult the network subscriber identity (e.g. IMSI) that matches the ARCADIAN-IoT ID that came attached to a given reputation score
- c. To generate reputation-based communication policies, understandable by Open5GS, according to the reputation scores received and the existent reputation policies; and forward these policies and the related network subscriber ID to the *Open5GS-Agent*
- d. To distribute authorization information to devices' security eSIM applet (for self-protection and self-recovery actions).

The second subcomponent of the Network-based Authorization has the purpose of communicating with Open5GS, to place the new subscriber policies in the repository for that purpose and perform the reset of a subscriber that has a new policy to be enforced in PCRF. In the Open5GS, having the information of policies to be enforced in the proper repository, the PCRF will assume the decisions over the communication flow accordingly, being them enforced in the PGW (Packet Data Network Gateway from Open5GS).

3.2.3.3.1 API specification

No external APIs exist in the Network-based Authorisation beyond the interfaces specified in Table 8. For details on the content received from the Reputation System please see Table 11 in section 3.3.3.5.2.

3.2.4 Evaluation and results

The current technical results can be found in Table 7. This component shows unit and functional tests, and integration tests with the eSIM security applet and with ARCADIAN-IoT publish/subscribe infrastructure (which will connect with the Reputation System). These tests were successful and, so far, the hypothesis of using PCRF / PCF for enforcing Network-based Authorization based on entities trustworthiness remain valid. The use of Open5GS as testbed for the network core elements necessary for the research also proved to be valid.

3.2.5 Future work

The envisioned future work focuses on building upon the current prototype to complete its functionalities according to the requirements and KPIs. The assessment made in the context of WP5 will inform the research direction as well. The envisioned future work is the following:

- Incorporate the final, more granular, set of trust-based policies, including authorization policies for the flow from internet services to devices.
- Research on the integration of real devices with the technologies of the network testbed.
- Assess the possibility of integrating with ARCADIAN-IoT's permissioned blockchain component for retrieving the reputation policies and scores (avoiding thus the current centralized database for the purpose).
- Perform the tests and evaluation of the final prototype.

3.3 Reputation System (UC)

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. In particular, different reputation algorithms are considered to build the score: a) the alpha-beta model; and b) 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- (example– 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 (M20)
300 or higher	10

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 (M20)
In the order of milliseconds	In the order of seconds

KPI scope	
Determine Reputation Score	
Measurable Indicator	
Number of entities supported in the reputation system	
Benchmarking (OPTIONAL)	
Not applicable	
Target value (M30)	Current value (M20)
3 entities	1 (devices)

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 algorithms 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 users⁵⁰.

Of particular interest in ARCADIAN-IoT is the consideration of the beta distribution^{51,52}, 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.

⁵⁰ 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

⁵¹ A. Josang and R. Ismail, "The beta reputation system," in Proceedings of the 15th, bled electronic commerce conference, vol. 5, pp. 2502–2511, 2002

⁵² 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.

The Dominance relationship-based reputation computation (DRBR)⁵³ 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, 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.
3. Considering the DAG determine the rank of services, considering the following equation:

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

Where r_i corresponds 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.

⁵³ 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.

Data Stream Processing Engines (DSPEs) like Apache Spark, Apache Flink, Apache Storm, Apache Samza⁵⁴ 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 – throughput, the possibility of using in-memory storage to parse, process data according to certain filters; 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.

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. 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 respect to the association of the reputation with policies, a simple approach is followed, inspired by the iptables⁵⁵ functioning mode. In a simplistic fashion there are default policies (ACCEPT or DENY) which apply to all the flows, with the exceptions that are specified for specific applications/services and endpoints. As a matter of example, The default policy can deny all the traffic, but exceptions may accept the traffic that is intended to the port 443 where the a web server provides content through the HTTPS protocol.

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

⁵⁴ 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>

⁵⁵ <https://www.netfilter.org/>

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 9 - 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.

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

The Reputation System is being designed to act as a Relying Party in remote attestation procedures. The research in this aspect has been initiated but has not yet solid results.

3.3.2.2 Research findings and achievements

The main research findings refer to the design and specification of the reputation system able to formulate a score for the reputation of an entity. The research performed also aimed to validate the reputation model, in particular the alpha beta model.

3.3.2.3 Produced resources

The produced resources include:

- The first version of the reputation system was initiated, implemented as a docker container using Java programming language.
- 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).

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 40.

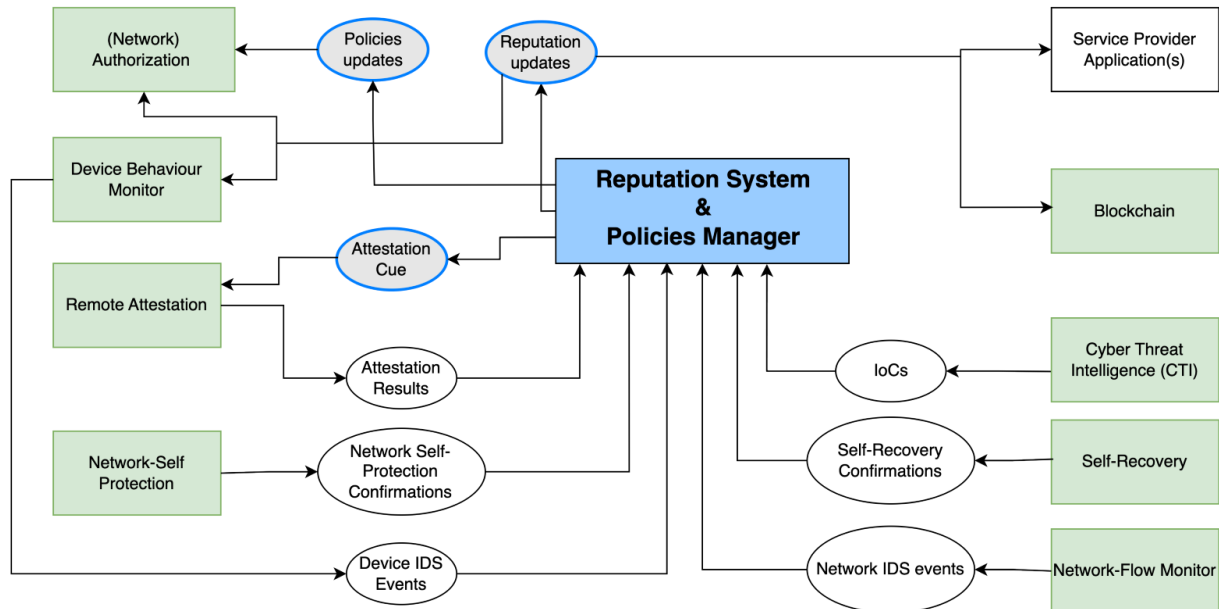


Figure 40 - Reputation System & Policy Manager logical architecture view

The reputation system receives information, from at least 6 ARCADIAN-IoT components and provides three types of information:

- Attestation Cue
- Reputation Updates
- 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.

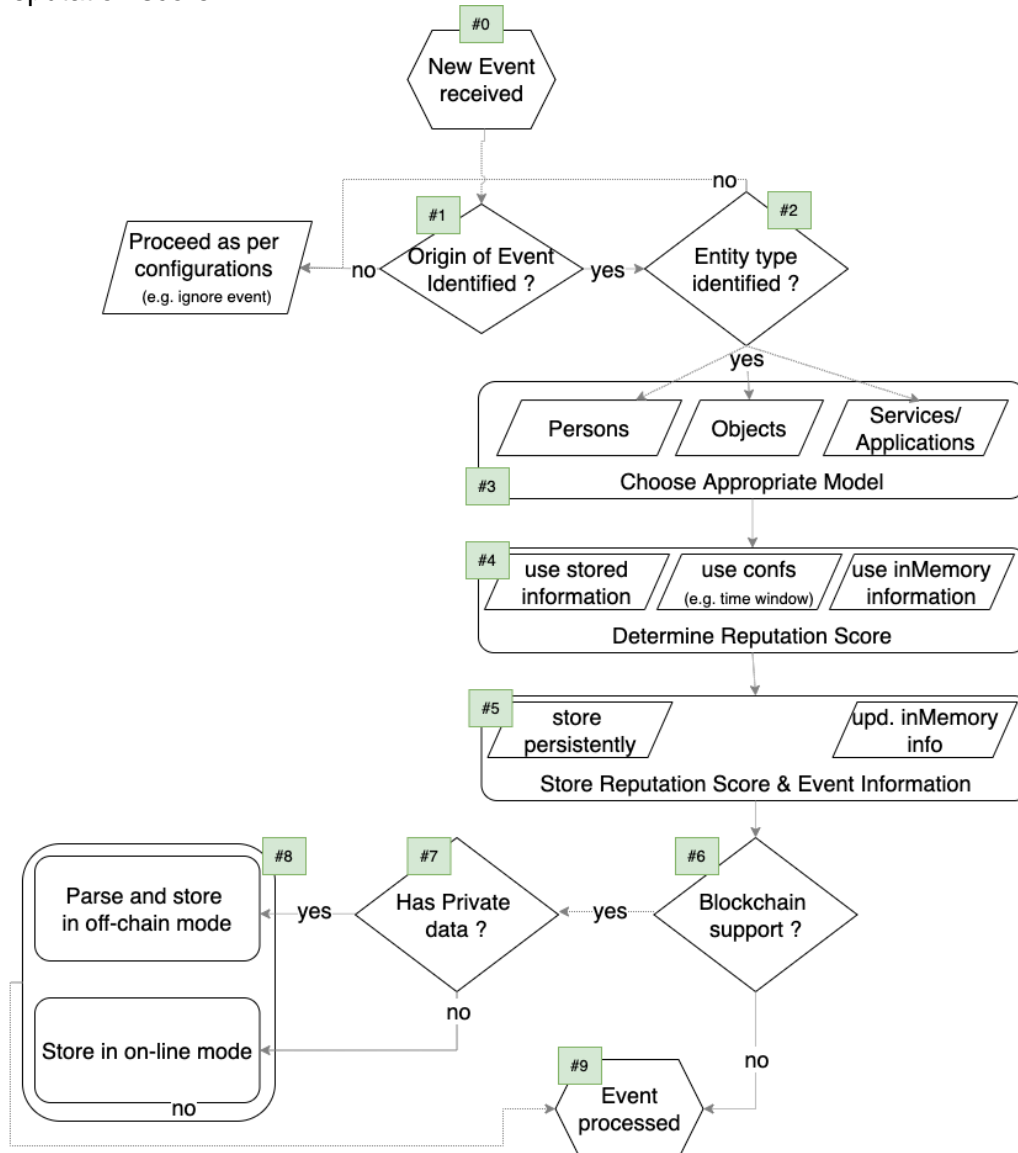


Figure 41 - Reputation System internal logic to determine reputation score

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 ask remote attestation,
- 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 (in prototype 1) and policies for attestation (prototype 2).

The reputation updates are also planned to be shared with the 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 10 - 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 11.

Table 11 - 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>, srcID: <AIoT Identifier> }</pre>
Policies Updates	<ul style="list-style-type: none"> • JSON format, and exchanged when there are CRUD operations in the policies • Information Items: <ul style="list-style-type: none"> • Active (mandatory) <ul style="list-style-type: none"> ○ Boolean representing whether the policy is active or not

	<ul style="list-style-type: none"> • reputationRange (mandatory) <ul style="list-style-type: none"> ○ Reputation range at which the policy is active ○ Range [min, max] • Description (optional) <ul style="list-style-type: none"> ○ Optional text description of the policy • Action (mandatory) <ul style="list-style-type: none"> ○ Policy effect (allow or deny) • SrcID (optional) <ul style="list-style-type: none"> ○ <u>Array</u> with IDs of the domain targeted by the policy ○ Can include: AIoT identifiers, • DstID (optional according to policy) <ul style="list-style-type: none"> ○ Array with IDs of the destination domain ○ Can include: AIoT identifiers, IP addresses, FQDN • DefaultPolicy (mandatory) <ul style="list-style-type: none"> ○ Boolean if true the policy is a default policy for instance to allow all the SrcIDs ○ If false it is a specific policy (same logic of iptables)
--	--

3.3.4 Evaluation and results

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

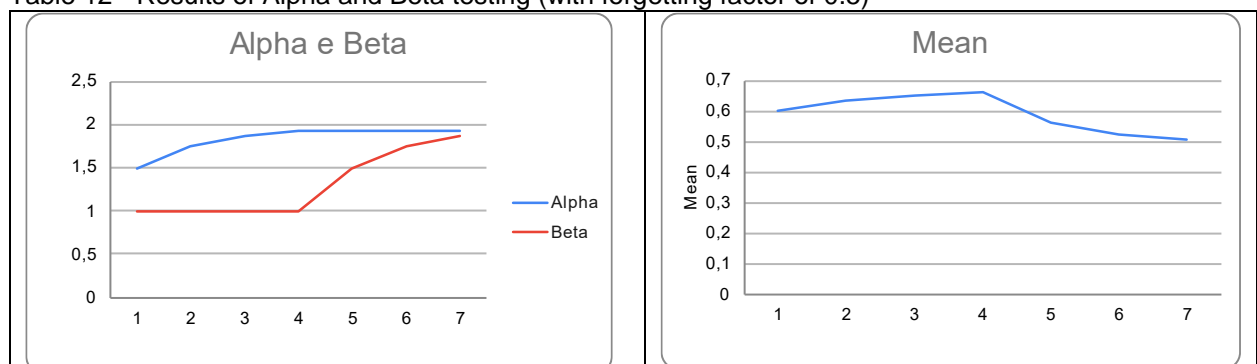
3.3.4.1 Experiment with Alpha-Beta model

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 12 - 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.

3.3.5 Future work

The upcoming activities will focus on the determination of reputation score for the events received from other components and for services in the domain use cases. The connection with the blockchain and the remote attestation modules will be pursued.

3.4 Remote Attestation (IPN)

3.4.1 Overview

3.4.1.1 Description

The Remote Attestation 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.

ARCADIAN-IoT aims to support Remote Attestation, with the ability to leverage Root-of Trust using a Secure Element – (e.g., eSIM, or crypto chip). The Remote Attestation component will consider the heterogeneity regarding devices' capabilities, targeting the support by both constrained IoT devices (e.g. drones, industrial devices) to less constrained ones like smartphones, being designed with efficient remote integrity verification and challenge-response mechanisms, while being aligned with the IETF Remote Attestation Procedures (RATS) working group⁵⁶, by both continuously monitoring its main progresses (with respect to standardized formats for describing claims and associated evidence, and procedures to deliver these claims) and opportunities for contributions.

The support of remote attestation for assessing trustworthiness in IoT services is also scoped. Namely, the integrity of services will be attested by appraising service-specific information from IoT devices (e.g., service configuration properties). This approach can enable obtaining proofs over the trustworthiness (and affect reputation) of both IoT devices (client perspective, e.g., smartphone in DGA service) and the service which the IoT device is supporting (server perspective, e.g., drone in DGA service).

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 – **Attestation pre-installation**: 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

⁵⁶ <https://www.ietf.org/archive/id/draft-ietf-rats-architecture-16.html>

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

KPI scope	
Novel RATS-based Remote Attestation	
Measurable Indicator	
Number of devices/OS platforms supported by remote attestation	
Benchmarking	
Not applicable	
Target value (M30)	Current value (M20)
At least 2	None

KPI scope	
Attribute-Based Encryption for Evidence	
Measurable Indicator	
Usage of Attribute-based encryption in evidence encryption	
Benchmarking	
Not applicable	
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	
Benchmarking	
Not applicable	
Target value (M30)	Current value (M20)
At least 1	eSIM supported via Hardened Encryption (but not tested / validated)

KPI scope	
Watchdog-based attestation triggering at Verifier	
Measurable Indicator	
Availability of a watchdog-based functionality to make sure the device is periodically attested	
Benchmarking	
Not applicable	
Target value (M30)	Current value (M20)
Available	Not available

KPI scope	
Support of Attestation Cues from Reputation System (for initiating new Remote Attestation processes)	
Measurable Indicator	
Availability	
Benchmarking	
Not applicable	
Target value (M30)	Current value (M20)
Available	Not available

KPI scope
Attestation Results feeding both device and service reputation models
Measurable Indicator
<ol style="list-style-type: none"> 1. Types of IoT devices reputation affected by Attestation Results 2. Number of IoT services reputation affected by Attestation Results

Benchmarking	
Not applicable	
Target value (M30)	Current value (M20)
1. At least 1	1. 0
2. At least 1 (among domains A, B and C)	2. 0

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 for 2018⁵⁷ are depicted in - OWASP's top 10 IoT security issues (2018 version). 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 reports. 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 ARCADIAN-IoT-compliant services.
2. **Insecure ecosystem interfaces:** attestation goes beyond simple IoT endpoint authentication and enables service providers to define policies and evidence 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; this proves as an attractive feature for service providers to enable access only to IoT endpoints which are up-to-date and known as secure.
4. **Insufficient privacy protection:** ARCADIAN-IoT's Remote Attestation ensures devices' evidence confidentiality via Hardened Encryption (itself supported by eSIM as RoT to sign encrypted evidence to ensure its integrity and trust).
5. **Lack of device management:** Remote Attestation is planned for usage by the Reputation System (as Relying Party). The Reputation System, upon the specific policies of the service provider (and its domain), will measure trustworthiness/reputation of IoT devices based on attestation results (and other information), affecting their authorization levels (e.g. leading to blacklisting).

⁵⁷ https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10

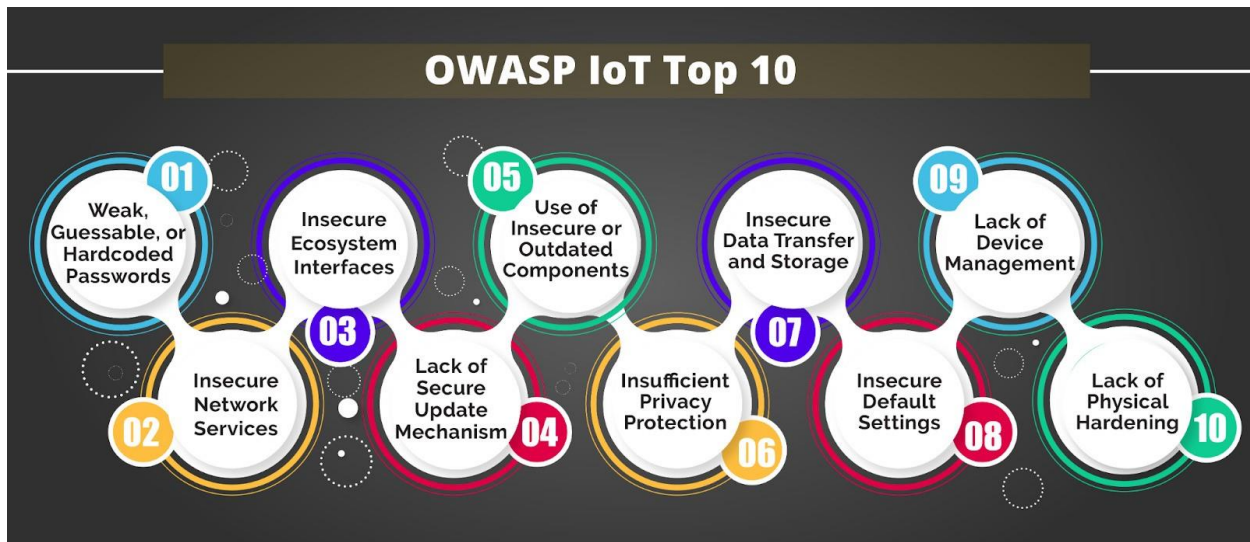


Figure 42 - OWASP's top 10 IoT security issues (2018 version)

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 8949⁵⁸. CBOR 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 protocol⁵⁹ that is used on CHARRA implementation of the IETF RATS model for Remote Attestation procedures.⁶⁰

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, we 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 survey¹ 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 as follows:

- **Remote Adversary:** the attacker can launch remote code 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

⁵⁸ <https://datatracker.ietf.org/doc/html/rfc8949>

⁵⁹ <https://en.wikipedia.org/wiki/CBOR>

⁶⁰ <https://github.com/Fraunhofer-SIT/charra>

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 are specifically against 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. Such attacks include:

- Proxy attacks;
- 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 against 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): an attacker is able to compromise an attester and removing associated evidence before attestation time;
- Return Oriented Programming (ROP);
- Data Oriented Programming (DOP).

The following are guidelines for mitigating attacks against discrete evidence collection:

- Randomness in the time interval between the moments where the challenge is sent and in the memory walk strategy.
- Evidence gathering compilation written 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).

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

The main research achievements refer to the design and specification of a novel Remote Attestation scheme with the following differentiating factors:

- eSIM-based hybrid Root of Trust (RoT) to sign the evidence, through Hardened Encryption
 - indirectly supported via integration with Hardened Encryption component; to be tested and validated;

- support of Attribute Based Encryption (ABE) for protecting evidence data confidentiality and enabling different entities (e.g. Verifiers) to access different sets of claims
 - o currently partially implemented;
- RATS-alignment, by adopting the CBOR format to encode and transmit evidence, attestation results, and reference values
 - o implemented and tested using dummy data;
- “Watchdog-based” periodic attestation triggering by the Verifier
 - o Not yet implemented;
- Support of “Attestation cues” from the Reputation System, as requests to the verifier to trigger an attestation process
 - o Not yet implemented

3.4.2.3 Produced resources

An intermediate implementation of the Remote Attestation system, referred to as RA2IoT, was initially produced, employing mbedTLS² library for cryptographic functions (encryption and decryption). This version was then evolved by replacing the mbedTLS library with Hardened Encryption’s ones. The implementation of RA2IoT is located in 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 the specific case of attestation evidence data transmitted by the attester. Furthermore, the result from the attestation process is processed by the Reputation System, affecting entities reputation according to pre-established manufacturer- or service-specific policies.

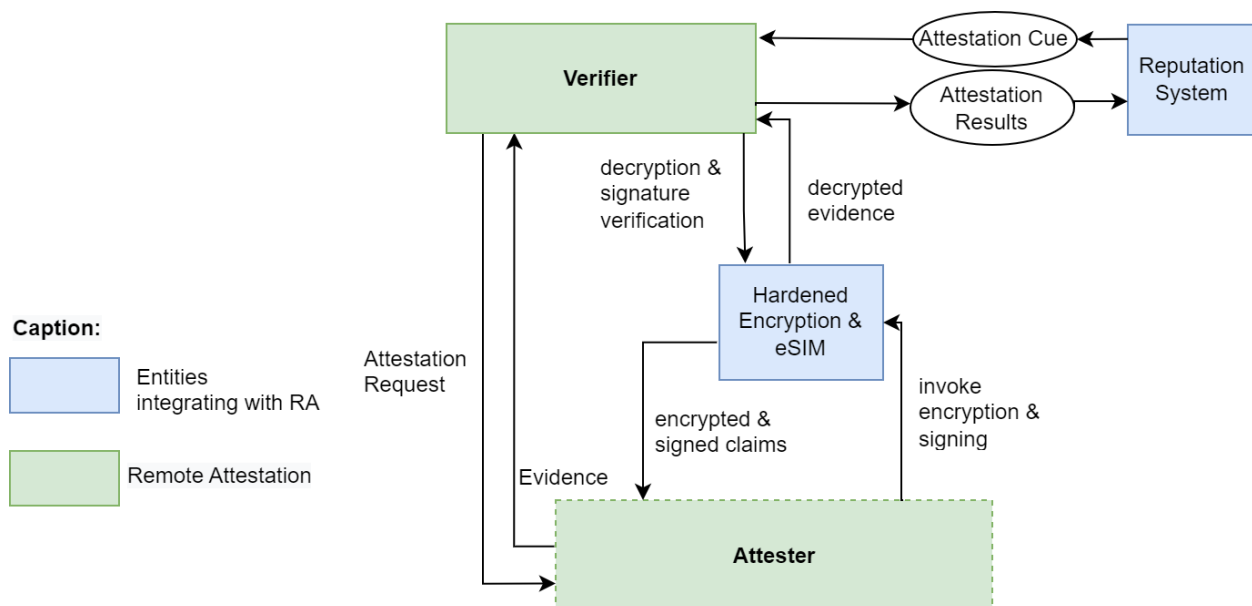


Figure 43 - Remote Attestation logical architecture and external dependencies

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 obtained from the manufacturer. Example use cases include A1,

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:

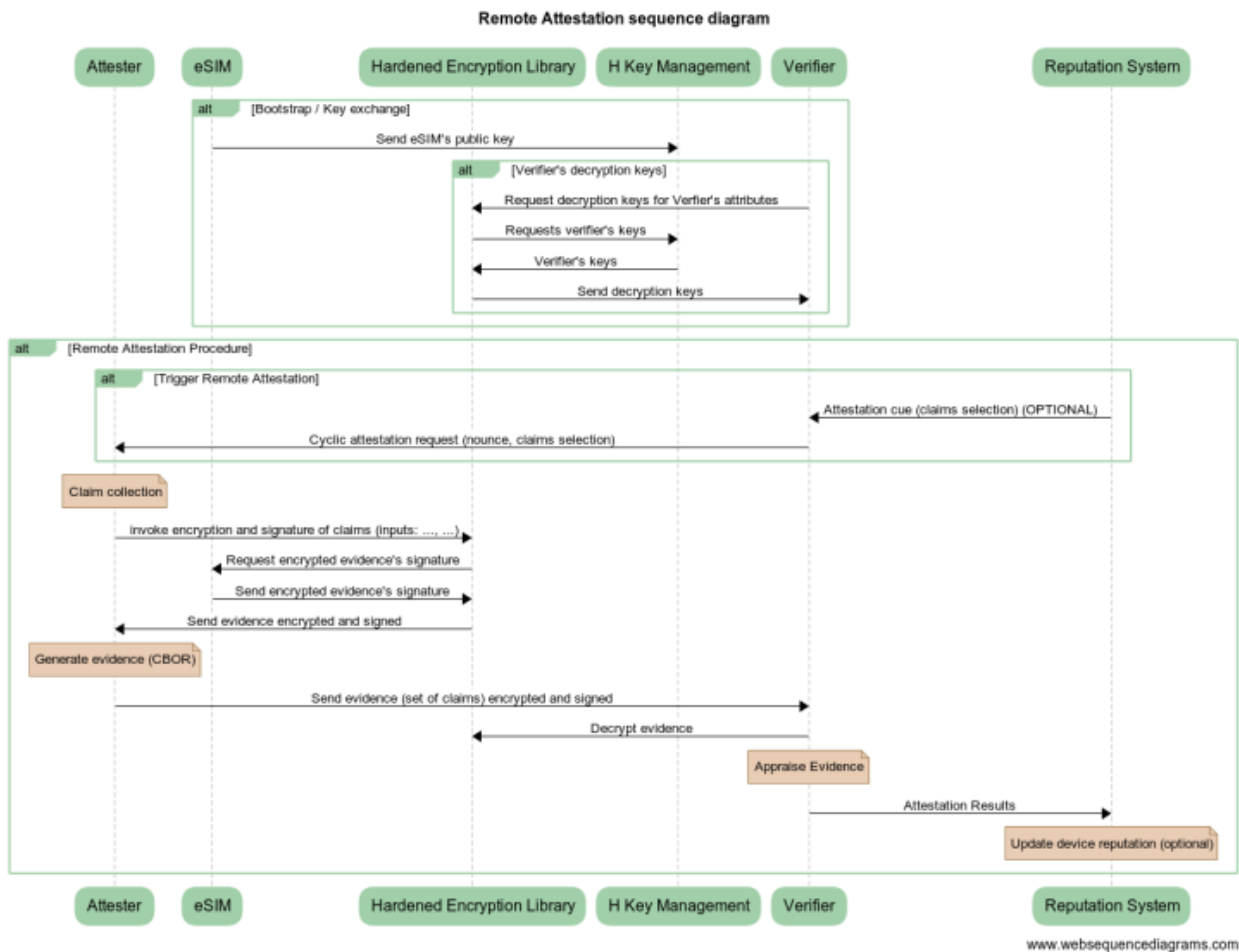
- a) Via an Attestation Cue from the Reputation System (e.g. upon sudden reputation decrease, event received from Behaviour Monitoring)
- b) Regular, watchdog-based attestation cycle, initiated by the responsible Verifier (which may either be responsible for appraising device, service-specific or both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

In this use case, remote attestation is used to appraise either devices' (e.g. HW model, build version) or service-specific (e.g. application fingerprint) claims, leading to multiple possible main scenarios:

- a) **Remote attestation successful:** all evidence has been appraised leading to attestation results passing the established evidence appraisal policies.
- b) **Remote attestation partially successful:** part of the claims have passed according to the established evidence appraisal policies.
- c) **Remote attestation failed:** all of the claims have passed according to the established evidence appraisal policies.
- d) **Invalid remote attestation response:** the response obtained by the Verifier contains critical issues (e.g. wrong nonce) which could be an indicator of attack attempt or intrusion.

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.

3.4.3.3 Sequence diagrams



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. The Constrained Application Protocol (CoAP) is used 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.

In each evidence transmission, 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.

3.4.3.4.2 External interfaces:

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.

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. The message bus is also used by the Reputation System to send Attestation Cues (for initiating remote attestation procedures upon its policies).

Verifier – Service Provider

Interface used by Verifier to collect Reference Values associated to ARCADIAN-IoT compliant services (e.g. DGA, Medical IoT). Similarly to the interface with the Reputation System, the information will be received via the message bus.

3.4.3.5 Technical solution

3.4.3.5.1 Deployment architecture view

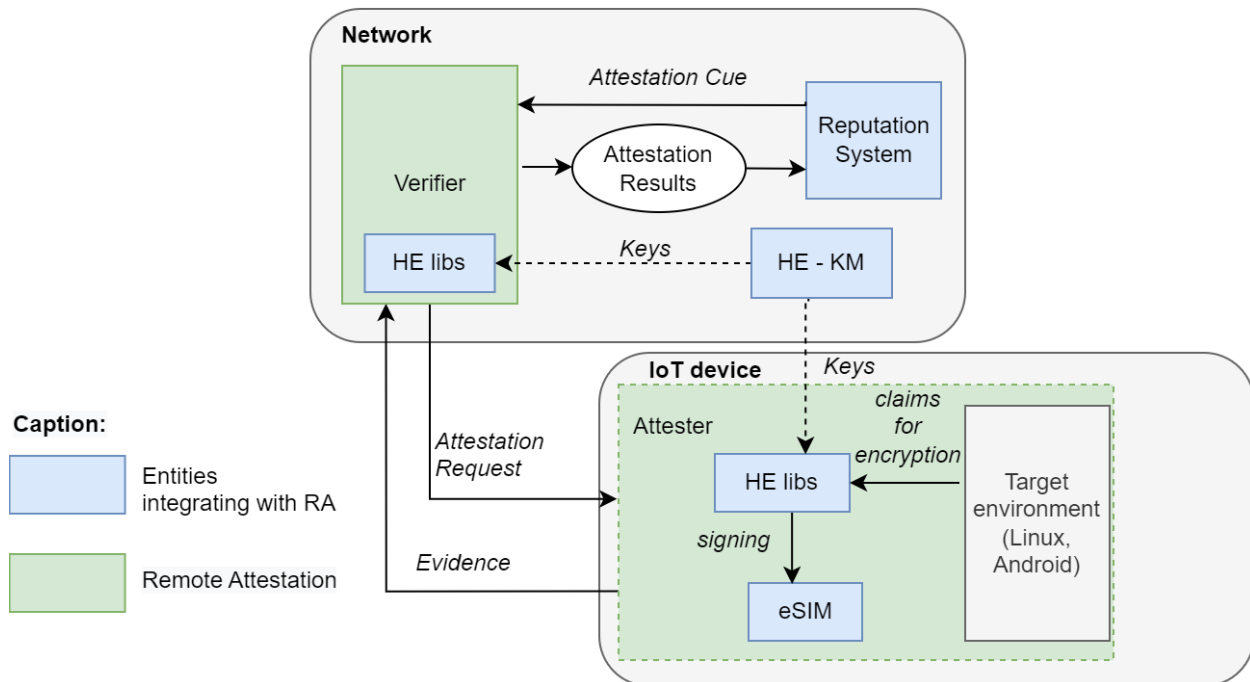


Figure 44 - Remote Attestation for ARCADIAN-IoT (RA2IoT) deployment architecture view

3.4.3.5.2 API specification

No APIs are provided by Remote Attestation. Information exchange with external entities will be done via the ARCADIAN-IoT message bus exchange `ra_exchange` for:

rReceiving Attestation Cues, through the `ra_exchange.attest_cue` topic;

Sending (device and service) Attestation Results (information exchange with Reputation System), through the `ra_exchange.attest_results` topic;

rReceiving service-specific Reference Values (information exchange with Service Providers), through the `ra_exchange.ref_values`.

3.4.3.5.3 Security aspects

IETF's draft on Reference Interaction Models for Remote Attestation Procedures⁶¹ 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*).

⁶¹ <https://www.ietf.org/id/draft-ietf-rats-reference-interaction-models-06.html>, section 4

The main endorser is the ARCADIAN-IoT's hardened encryption component. Since the attester uses this component's libraries 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 is able to (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 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 following list of claims is currently being considered for the smartphone as less constrained IoT devices (in domains A and C):

- a. **Product:** A value chosen by the device implementer containing the device's development name or code name.
- b. **Device:** A value chosen by the device implementer identifying the specific configuration or revision of the device's body (sometimes called "industrial design").
- c. **Board:** A value chosen by the device implementer identifying the specific internal hardware used by this device.
- d. **Version release:** The version of the currently executing Android system.
- e. **ID:** An identifier chosen by the device implementer to refer to a specific release. In the project, each device will be attributed a uniquely identifiable ID by the ARCADIAN-IoT framework.
- f. **Version Incremental:** A value chosen by the device implementer designating the specific build of the currently executing Android system. A typical use of this field is to indicate which build number or source-control change identifier was used to generate the build.
- g. **Brand:** A value chosen by the device implementer identifying the name of the company, organization, individual, etc. who produced the device.
- h. **Android version/API Level** - Applications can use a manifest element provided by the framework API — `<uses-sdk>` — to describe the minimum and maximum API Levels under which they are able to run, as well as the preferred API Level that they are designed to support.⁶² The SDK level (integer) the phone is running is available in `android.os.Build.VERSION.SDK_INT`.
- i. **Device State** – Check if the device state is LOCKED – prevents from flashing new software to the device, verification is enforced – or UNLOCKED – allows modification.⁶³

As for the **Drone** (in domain A), the following claims will be considered:

- a. **OS Release**
- b. **OS Version**
- c. **Hardware – “uts.machine”**
- d. **Device state**
- e. **GPS location**

⁶² <http://www.dre.vanderbilt.edu/~schmidt/android/android-4.0/out/target/common/docs/doc-comment-check/guide/appendix/api-levels.html>

⁶³ <https://source.android.com/security/verifiedboot/device-state>

As for claims in the context of industrial devices, it is still subject to analysis, as the support of RA2IoT in the smart grid use cases (domain B) is unclear.

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:

- A-aiotID of the device being attested
- The result of the attestation response’s signature verification (a Boolean value).
- The result of the nonce’s verification (a Boolean value).
- Information related with the validation of the claims

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

At this point, as mentioned in the achievements and as a result of the implementation of the attester and verifier functionalities, including the support for CBOR for evidence encoding and transmission, the ARCADIAN-IoT remote attestation procedure has been partially validated, considering dummy data (thus agnostic to any particular environment). The support of ABE for Remote Attestation has been validated after successful integration with the Hardened Encryption, however additional tests are necessary (e.g. involving 2 different Verifiers in the same use case). The effective support (demonstration) of eSIM-based hybrid RoT will be validated once testing has moved to real environments / devices.

3.4.5 Future work

The upcoming activities will be focusing on 1) the support of real execution environments (smartphones and drones), 2) specifying and implementing approaches for remote attestation of services running in IoT devices, and establishment of approach for enabling multiple verifiers (e.g. Verifier for device attestation according to device manufacturer policies, Verifier for service attestation according to service provider policies); 3) the definition of policies for translating attestation results into entities (devices and services) reputation updates and implementing the interface (via ARCADIAN-IoT’s message bus) with the Reputation System for receiving Attestation Cues and transmitting Attestation Results; 4) the research and implementation of approaches for enhancing the remote assessment of the device’s integrity and trustworthiness of the measurements it provides.

4 RECOVERY PLANE

4.1 Self-recovery (XLAB)

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 constrains may render the encryption process unfeasible. Addressing this issue will be an encryption proxy that is able to receive plain data, encrypt it and either returning 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 platform only stores backup metadata.

4.1.1.2 Requirements

A recall of the requirements defined in WP2 with further supplemental information as needed. Further clarifications of existing requirements and/or new requirements, can be detailed here.

Requirement 7.1.1 – Recovery mechanism

Each recovery system requires first an organised and detailed description of a running system. On 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, set of scripts that set up the processes and the machine which can run the recovery process and has access to the services and/or infrastructure that require recovery (network connectivity, etc).

In addition to the requirements 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 (M20)
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 (M20)
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

XtreamFS⁶⁴ 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

⁶⁴ <http://www.xtreemfs.org/>

it untenable as a choice in a modern platform, mainly the use of older versions of dependencies 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, CEPH⁶⁵ 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 will move to OpenAPI v3 and the server will be rewritten in Golang.

4.1.2.3 Produced resources

The first prototype version of the Self-recovery component is available on the project's Gitlab repository⁶⁶. It includes the server component, client-side scripts for performing the recovery operations in Bash 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

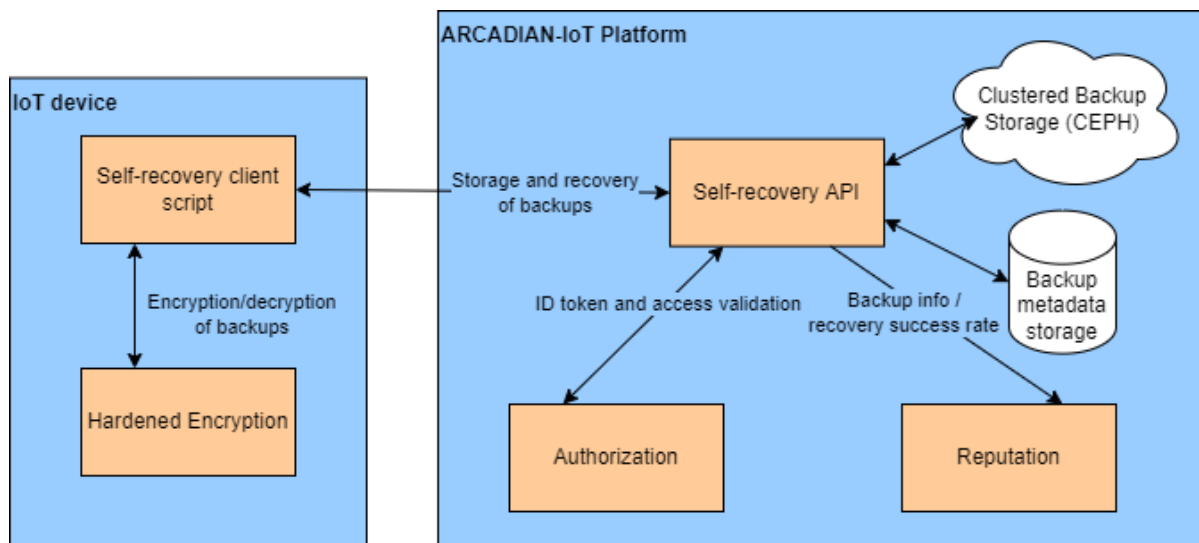


Figure 45 - Self-recovery logical architecture view

Self-recovery comprises client-side (on-device) and server modules, where the backups and related metadata are stored. Figure 45 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

⁶⁵ <https://docs.ceph.com/en/quincy/>

⁶⁶ https://gitlab.com/arcadian_iot/self-recovery/-/tree/develop

a CEPH cluster, while that backup metadata (size, time, signature etc.) are stored in a standard relational database.

4.1.3.2 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. 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 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.

4.1.3.3 API Specification

The API specification of the Self-recovery server module is described in Swagger OpenAPI format in a YAML file residing on the Gitlab repository⁶⁷ of the project. The API specification is still in its development stage and will be expanded and refined during the coming implementation phase. One of the expansions is adding an endpoint that allows users to generate a recovery key and store it as a QR code. This process assists users in recovering their credentials from an out-of-band backup of their wallet or other identities when they are unable to authenticate with the ARCADIAN-IoT platform.

The proposed format for this new endpoint is the following:

⁶⁷ https://gitlab.com/arcadian_iot/self-recovery/-/blob/develop/server/api/swagger/swagger.yaml

Endpoint	GET /api/recovery_key
Response	<pre>{ "key": "afc5ce76-898c-415f-ae8a-754c4602a2e6", // UUIDv4 placeholder, will be updated with a more suitable mechanism "qr_code": "iVBORw0KGgoAAAANSUHEUgAAASwAAAEsCAYAAAB5fY51AAAABGdBTU EAALGPC/xhBQAAACBjSFJNAAB6JgAAgIQAAAPoAAACA6AAAdTAAAOpgAA A6mAAAF3CculE8AAAABmJLR0QAAAAAAAAAD5Q7t/AAAACXBIWXMAAAABgAA AAYADwa0LPAAAGjUIEQVR42u3dQW4qRxRAURNIB5a9//V54C2QQfSHCZbK xavbnDMHmgZd1eDp9e1+v9/fAAL+mr4AgJ8SLCBDsIAMwQlyBAViECwgQ7CA DMECMgQLyBASiEOwgAzBAjIEC8gQLCBDsIAMwQlyBAViECwgQ7CADMEC MgQLyBASiEOwgAzBAjIEC8gQLCBDsIAMwQlyBAViECwgQ7CADMECMgQLy Ph7+gL+eH9/f/v+/p6+jG3u9/vW97/dbkufv/v1q06/f/6/z+GEBWQIFpAhWECGYA EZggVkBQaQIVhAxjFzWI98fX29fXx8TF/Gf1qdw1mdA1o1Pae1e87r0fvvvr9X//8 +ixMWkCFYQIZgARmCBWQIFpAhWECGYAEZmTmsR+r7mFbff/X1V5+z2v3+q +r/32dxwglyBAViECwgQ7CADMECMgQLyBASiOMyc1h1p8/hrF7f7n1f03NgPlcT FpAhWECGYAEZggVkBQaQIVhAhmABGeawDrF739Tq51/dq3//CicsiEOwgAzB AjIEC8gQLCBDsIAMwQlyLjOHVZ+2b2Pafq5fKfv+5pWv/5nccICMgQLyBASiEO wgAzBAjIEC8gQLCAjM4f1+fk5fQmjdu/Lmn7uYP31j7z6//e3OGEBGYIFZAgWkC FYQIZgARmCBWQIFpBxu1vEk2Cf1P9zf16DExaQIVhAhmABGYIFZAgWkCFY QIZgARnH7MPaPUezavc+qatf3/RzEVfvz/S+renXn8IJC8gQLCBDsIAMwQlyBAV iECwgQ7CAjGPmsB6ZnhM5fQ5m+rmEu+ekdr++fn2rKnNaTlhAhmABGYIFZAg WkCFYQIZgARmCBWRk5rCm56Cuvq9o976oVdPPHZz+fU//zyLExaQIVhAhm ABGYIFZAgWkCFYQIZgARmZOazpOZLdcy7Tr5/ehzX93MTV71ef01r9/s/ihAVk CBaQIVhAhmABGYIFZAgWkCFYQMYxc1inz/HsnnPZbfec11XmfHZ9v+nPP/3+/ pQTFpAhWECGYAEZggVkBQaQIVhAhmABGbf7IQMau/dNTc+xtO9LenXTc3r8 DicsiEOwgAzBAjIEC8gQLCBDsIAMwQlyjtmHNW16zmq30+fQdjv9/k/vW5u+Pz/l hAVkCBaQIVhAhmABGYIFZAgWkCFYQEZmH9Yj03NC05+/yhzaXtPPdZz+f/8 WJywgQ7CADMECMgQLyBASiEOwgAzBAjly+7Cm9wU9snsO6fQ5r93PVVx9/f Sc0vT/85Q5qIVOWECGYAEZggVkBQaQIVhAhmABGYIFZGTmsB65+j6n3dc/P Sf0yPT9r89x1f/ffzhARmCBWQIFpAhWECGYAEZggVkBQaQcZk5rGnTc0y792 3t/n7T+7pWr2/3HNPufWEVTlhAhmABGYIFZAgWkCFYQIZgARmCBWRcZg5r9 xzK9PtPf/4jp8/57N4X9urPXXwWJywgQ7CADMECMgQLyBASiEOwgAzBAjJu90 MGLKbnjKb3Qe22ew6oPudT/31XT..." // base64 encoded jpg of QR code }</pre>

It is important to note that the Self-recovery server offers the functionality to generate a recovery key, but does not store it, it only returns the contents to the user.

4.1.4 Evaluation and results

A first prototype version of the Self-recovery component has been produced. This version is mostly stand-alone but has already been integrated with Hardened Encryption. The backup and recovery process, including encryption and decryption of backed up files, has been tested and validated in a simulated environment with dummy data.

4.1.5 Future work

Development towards the second prototype version of Self-recovery will focus on completing integration with the other ARCADIAN-IoT components. The server will be expanded with the capability to connect and communicate through an AMQP system (RabbitMQ), facilitating inter-

component communication and integration, in the case of Self-recovery component, the primary target for integration is the Reputation system.

The server rewrite into Golang will facilitate better integration with Hardened Encryption. Currently, the client-side integration with HE is already done, but the server is not capable of attesting the validity of the encryption of the stored backups. This capability will be achieved with the inclusion of the GoFE⁶⁸ library, that is already used by HE.

The use of CEPH will also be expanded to a properly clustered deployment, as the requirements to demonstrate the functionality of using a networked storage system by the Self-recovery component in the first prototype phase was only a single-node deployment.

⁶⁸ <https://github.com/fentec-project/gofe>

4.2 Credentials recovery (ATOS)

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 as there the trust between the device and the backend services is established.

It is proposed to provide authenticated and authorised access to the backup server for data based on the ARCADIAN-IoT ID Token described under the authentication component in section 2.4. This relies upon eSIM Network Identity (see section 2.2), Self-Sovereign Identity (see sections 2.1 and 3.1) and Biometrics (see section 2.3) with the latter only supported for persons. Therefore, it is these credentials that are under the scope of being able to be recovered.

To avoid manual recovery of the credentials, various techniques will be evaluated and the most suitable one will be implemented.

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**
 - o 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	
Benchmarking (OPTIONAL)	
-	
Target value (M30)	Current value (M20)
1	0

KPI scope	
Availability of self-recovery and decentralized identity management schemes.	
Measurable Indicator	
Support recovery of Decentralized Identifiers and Verifiable Credentials	
Target value (M30)	Current value (M20)
Recovery supported	Recovery not supported

4.2.2 Technology research

4.2.2.1 Background

ATOS do 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 the more secure option so that no unencrypted credentials are shared outside of the wallet whatsoever.

In the following sub-sections, we outline the approach to be 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 device is attempting a recovery of its credentials and later its data. Here, it is considered the recovery of the credentials in an automatic way before the data back-up can securely accessed. Note that if the credentials were otherwise compromised it would be needed to re-issue the credentials themselves as is described in sections 2.1 and 3.1.

The credentials that are backed up in Self-recovery should make use of Hardened Encryption cipher techniques to encrypt the back-up data. Ideally this is done at source but depending on the device capabilities may need a proxy solution.

Support for automatic 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, for a user's SSI Wallet the following mechanisms can be considered:

- during account registration in Self-recovery, it 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.
- during account registration in Self-recovery a user can be issued with a QR Code containing a recovery key to be used to request a restoration of the SSI Wallet including its Verifiable Credentials.

The SSI Wallet backup is encrypted, and the restoration process described above provides access to the key to decrypt it and restore the wallet. The encryption mechanisms that will be supported will be provided by Hardened Encryption component.

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 eSIM on a mobile/personal device

The root cause that could trigger the eSIM credentials recovery is a device (IoT device or personal device) being stolen or lost.

If this device is recovered (its owner has it again without being irreparably shattered), the eSIM credentials continue there, in the secure element. The threat surface related with the access to the eSIM 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 – eSIM 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, we will not question the state of the art of the SIM/eSIM secure element (UICC/eUICC) as being secure, and just consider the case that the device is not recovered by its owner.

In this case, a new device will have a new eSIM profile (new subscriber credentials) and the credentials recovery will consist of associating these new credentials with the previous ARCADIAN-IoT ID. This process will be similar to the onboarding process in what concerns the eSIM credentials association with an ARCADIAN-IoT ID.

Biometric Recovery

The biometric credential is the user's face which is not treated as a recoverable credential, as it is a physical trait of the person.

4.2.2.2 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 to the IoT Device to perform 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 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, the IoT Device is not seen to be issued with so many different credentials as a person would be to their wallet from many different organisations, so it should be considered if the effort here is worthwhile as it could be simpler to reboot the device swiping their one or two credentials and re-issuing them.

Credentials Recovery for eSIM on a IoT device

The credentials recovery for the cellular network subscriber, being it a IoT device or personal device should be the same, and only apply when devices' hardware is not recovered. Upon a recovery process, a new eSIM profile is provisioned to the substitute device and the new credentials are associated with the ARCADIAN-IoT entity being recovered.

4.2.2.3 Research findings and achievements

The initial research findings propose the import and export of encrypted back-ups of the wallet credentials based on a recovery key for a previously registered entity which can be either obtained from a quorum of trusted entities or a QR Code kept in a safe place.

The recovery of IoT Devices credentials can be based also on a recovery key issued for registered IoT Devices, with the recovery key being kept by a controller of the IoT Device, which would be responsible for sharing the recovery key when needed.

4.2.2.4 Produced resources

There are no existing resources supporting Credentials Recovery. Person credential recovery implementation and integration is planned first, although not in time for the first prototype P1. IoT Device credential recovery design, implementation and integration is also planned for the final prototype P2.

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 captures the 3 main sub-use cases for supporting the recovery of person credentials from a user's SSI wallet.

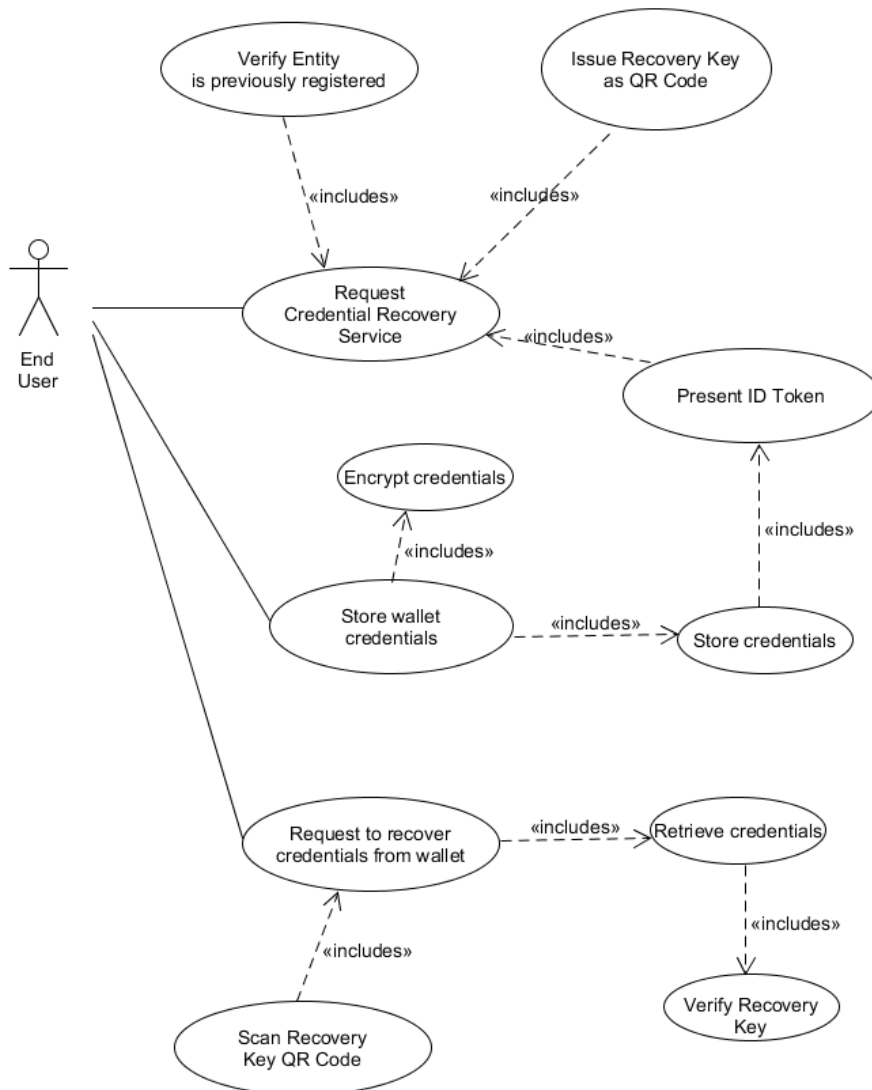


Figure 46 - Recovery of SSI Wallet Credentials Use Case

4.2.3.2 Logical architecture view

4.2.3.2.1 SSI Credential Recovery

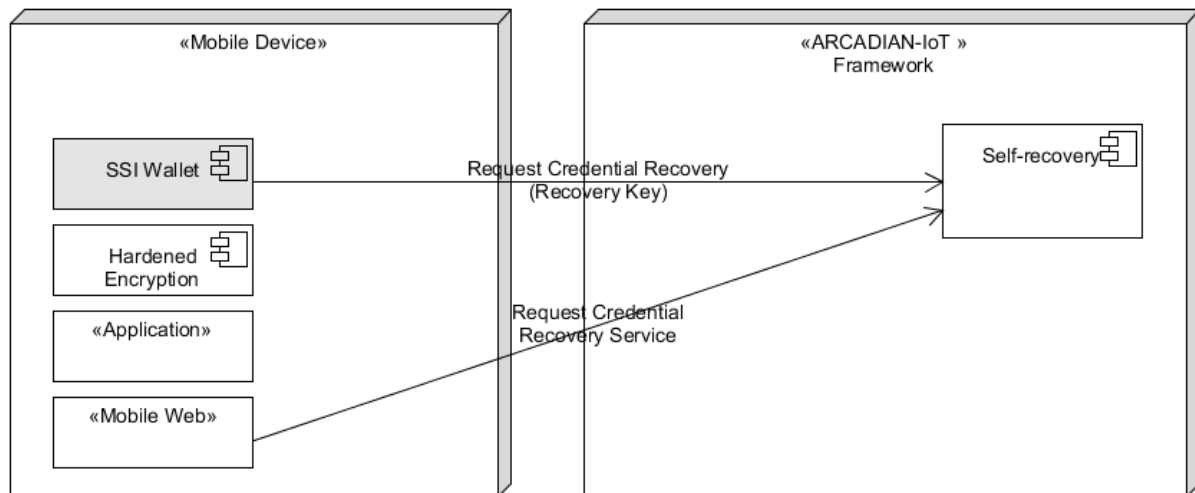


Figure 47 - SSI Credential Recovery Logical Architecture

4.2.3.3 Interface description

The mobile SSI Wallet will support an out-of-band interface by scanning the QR Code to obtain the recovery key.

To store and later retrieve the credential back-up, the wallet will call the Self-recovery component as described in section 4.1.

4.2.3.4 Technical solution

4.2.3.4.1 API specification

There is no API Specification on the SSI Wallet for Credential Recovery as the QR Code is scanned manually in an out-of-band process.

The store / retrieve credential back-up calls are provided by the Self-recovery API specification (see section 4.1.3.3).

4.2.3.4.2 Frontend design

The SSI IdP frontend will provide an interface to the user to request to Self-recovery to register for the Credential Recovery service and display the QR Code retrieved from Self-recovery. Note the user needs to be registered in ARCADIAN-IoT before they can register for the framework's Credential Recovery service.

4.2.3.4.3 Ledger uSelf mobile SSI wallet

The Ledger uSelf mobile SSI wallet will support:

- the symmetric encryption of the SSI credentials using the Hardened Encryption libraries.

- the storing of encrypted credential backups to Self-recovery.
- the scanning of the QR Code to retrieve the recovery key to retrieve the backup from Self-recovery.

4.2.4 Evaluation and results

The low-level design and implementation is ongoing with no results currently available.

4.2.5 Future work

The next step is to finalize the design and start implementation for client recovery of person credentials and integrate it with the Self-recovery component. Additionally, for IoT Device Credential Recovery it is needed to fully agree the design specification for its implementation in the final prototype P2.

5 CONCLUSIONS

This report has aimed at presenting the up-to-date research status for ARCADIAN-IoT's Vertical Planes – Identity, Trust and Recovery planes, covering both background, research and specifications, implementation and preliminary evaluation (or associated plans). Moreover, a list of revised (e.g. more aligned with the research goals) KPIs for each component have been provided. Most of these KPIs depend on actual validation in the domain pilots, which is yet to take place, but some of the KPIs could potentially be already interpreted as having been achieved, as their implementation is compliant with the target environments (e.g. operating system), only lacking the validation to be performed in the next couple of months.

A significant part of the research and specification activities have been completed, with a few notable exceptions (e.g. support of remote attestation with cryptochip as secure element or support of DID/SSI in industrial constrained devices). As regards the implementation / development, activities for most components are ongoing with the focus on diverse threads such as the registration of the different types of entities (persons, devices and services) or compatibility with the device heterogeneity (e.g. drone, smartphone, industrial gateway).

An overview of the ARCADIAN-IoT vertical planes research activities now follows. Within the **Identity Plane**, the different options for supporting **Decentralized Identifiers** are either already under implementation or require further research, in order to cope with the different use cases' needs. The **eSIM** has been target of enhancements for enabling network-based authentication of IoT persons and devices and next steps will focus its validation in different IoT services (and associated devices), while **Biometrics** is targeting more accurate and faster face verification of persons under challenging conditions. Moreover, a new Multi-factor authentication scheme for orchestrating the aforementioned authentication factors has been specified and partially implemented for enabling **Authentication** of persons and devices when accessing ARCADIAN-IoT-compliant IoT services.

The **Trust Plane**, paramount to enable a Chain of Trust (CoT) between the different entities (i.e. persons, devices, services), has been partially implemented, with focus on person-devices, person-services and device-services interactions. **Verifiable Credentials** are being considered for enabling trusted entities' identification, and are under final interwork specification for issuing, registration and authentication of IoT devices. **Reputation System** models entities' trustworthiness based on the interaction events involving the different entities, with next steps focusing effective determination of reputation scores based on the events consumed from operating IoT services and other ARCADIAN-IoT components. **Remote Attestation** capability for collecting hardened evidence for assessing IoT devices and services integrity has been partially implemented, with next steps including refining its role in determining entities reputation. **Authorization**, aimed at enforcing trust-based policies in the mobile network core and informing devices' secure element of the associated device trustworthiness is on track to achieve the target objectives, and, among others, will focus the incorporation of more granular trust-based policies and validation in the target IoT domains.

Finally, as for the **Recovery Plane**, a first stand-alone prototype of **Self-recovery** has been tested and validated for backup and recovery of dummy data, including encryption and decryption of backed up files, and next steps will focus stronger integration with ARCADIAN-IoT (e.g. Reputation System). The **Credentials Recovery**, necessary for establishing the trust between a device and the IoT service, is aiming at the recovery of credentials for the different ARCADIAN-IoT authentication factors, and will pursue the support of client recovery of person credentials and integration with Self-recovery, and the clarification of how to support IoT device credential recovery.

The outcomes of this deliverable are an integral part of the first pilot integration of ARCADIAN-IoT framework (P1), delivered in the scope of WP5. The final deliverable (D4.3) will provide a report on the final prototype implementation of the Vertical Plane components and their evaluation.

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	<u>Device</u> Reputation Ratings Values (min 0.0, max 1.0)	Service/Middlew are 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 NUL; (+) 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		-		-	



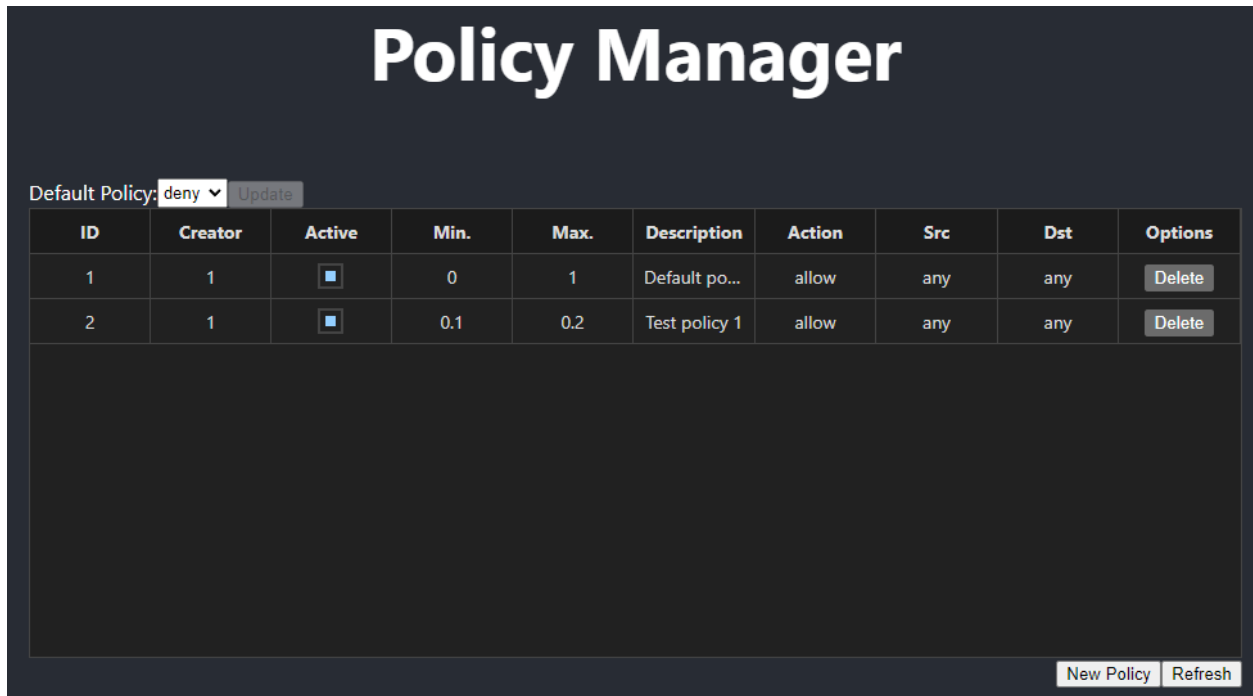
Appendix D – Policy Manager User Manual

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.



Default Policy: deny ▼ Update

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 checked="" type="checkbox"/>	0.1	0.2	Test policy 1	allow	any	any	Delete

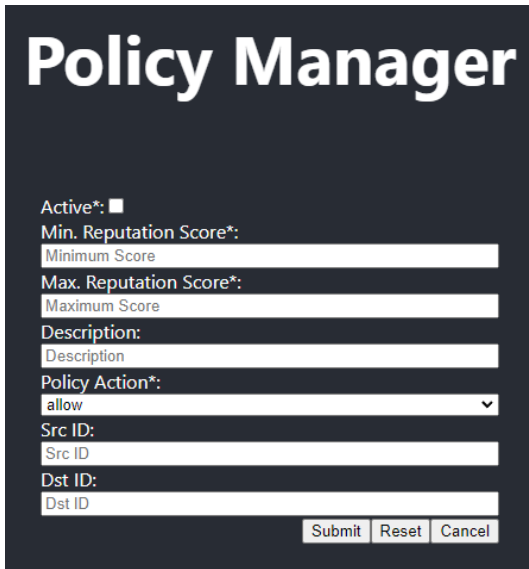
New Policy Refresh

Figure 48 - 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 49 – 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	BTCR DID 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:jlinc:	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
did: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/mattrglobal>, 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

