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

ARCADIAN-IoT

Autonomous Trust, Security and Privacy Management Framework for IoT

D4.1: ARCADIAN-IoT Vertical Planes

Revision: v.1.0

Work package	4
Task	Task Number 4.1, T4.2, T4.3
Due date	30/4/2022
Submission date	30/4/2022
Deliverable lead	ATOS
Version	1.0
Partner(s) / Author(s)	ATOS: Ross Little, Miguel Angel Mateo Montero

	IPN: Gabriela Bento, Paulo Silva, Sérgio Figueiredo UWS: Jose M. Alcaraz Calero, Qi Wang, Ignacio Martinez Alpiste, Gelayol Golcarenarenji, Julio Diez Tomillo, David Tena Gago, Javier Saez Perez TRU: João Casal, Carlos Morgado, José Rosa, Tomás Silva, Ivo Vilas Boas XLAB: Jan Antić

Abstract

This public technical report constitutes the deliverable D4.1 of ARCADIAN-IoT, a Horizon2020 project with the **grant agreement number 101020259**, under the topic **SU-DS02-2020**. D4.1 has the purpose of reporting on the initial investigations regarding the development of the ARCADIAN-IoT Vertical Plane components.

Keywords: ARCADIAN-IoT, Identity, trust, recovery, decentralized, eSIM, biometrics, verifiable, credentials, attestation, reputation, self-recovery

Document Revision History

Version	Date	Description of change	List of contributor(s)
0.1	10/03/2021	Updated ToC with all Vertical Plane components included	ATOS
0.2	18/03/2022	Overview section of each component added	UWS, TRU, IPN, ATOS, XLAB
0.3	11/04/2022	First draft of deliverable including full input for each component	ATOS, IPN, TRU, UWS, XLAB
0.4	14/04/2022	Format revision and bibliography	ATOS
0.5	26/04/2022	Updated after IPN review comments	ATOS, IPN, TRU, UWS, XLAB
0.6	29/04/2022	Final check and accepting updates and removing comments.	ATOS
1.0	29/04/2022	Final review and edits	IPN

Disclaimer

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

This is the first deliverable of ARCADIAN-IoT WP4, which is responsible for developing the **Vertical Planes of ARCADIAN-IoT Framework.** This technical report combines the initial analysis and investigations on how to provide each of the Vertical Plane components, taking into consideration the framework requirements identified in the D2.4 [1] and also the applicable domain use cases specified in D2.2 [2].

The vertical plane groups its components into sub-planes where they contribute towards the same objective, either in a complementary way or as rival technology that may be more applicable to certain domain use cases. The sub-planes and their 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 Pane (Task 4.3)**
  - Self-recovery
  - Credential recovery

The main outcome of this deliverable considers the State-Of-The-Art in the area of each of the Vertical Plane components, identifying technology and innovative approaches and describing also any existing resources that partners may have in this area that can be built upon to support the ARCADIAN-IoT requirements and use cases. Finally, the report considers the future work that is to be taken towards the development of each component.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

5GC	5th Generation Core
5GS	5th Generation System
AI	Artificial Intelligence
AIoT	ARCADIAN-IoT
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
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
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
PCC	Policy and Charging Control
PCF	Policy Control Function
PCEF	Policy and Charging Enforcement Function
PCRF	Policy and Charging Rules Function
RATS	Remote Attestation Procedures
REST	Representational State Transfer
RFC	Request for Comment
RoT	Root of Trust
SAS	Security Accreditation Scheme
SE	Secure Element
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 Objectives

The ARCADIAN-IoT framework includes vertical planes devoted to identity, trust and recovery management, which are supported by horizontal planes managing privacy of data, security of components and decentralized storage through blockchain technologies, as illustrated in Figure 1 below.

The objective of this deliverable is to report on the initial investigations regarding the development of the ARCADIAN-IoT Vertical Plane components that have taken place between month 6 and 12 of the project.

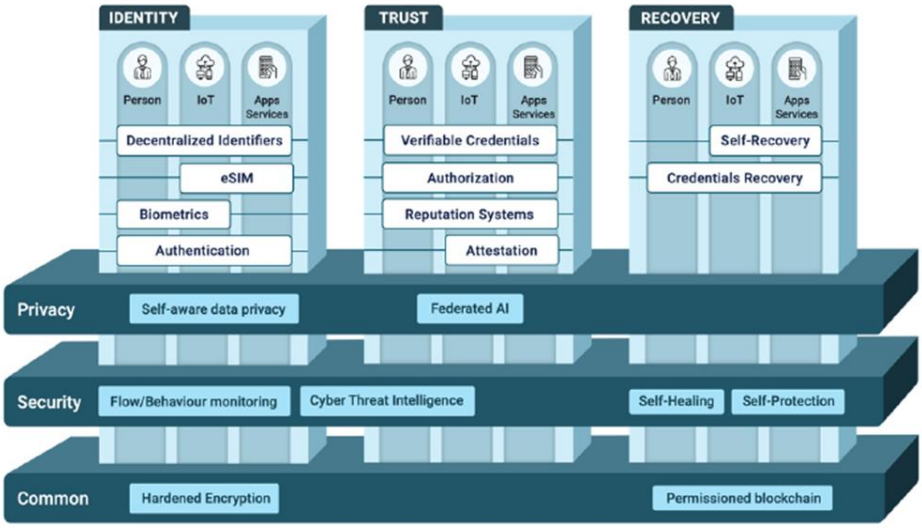

Figure 1 - The ARCADIAN-IoT framework

The Vertical Plane components under investigation in this deliverable are separated into the following sub-planes:

ARCADIAN-IoT Vertical Sub-Planes	Task
Identity management	Task 4.1
Trust Management	Task 4.2
Recovery Management	Task 4.3

The structure of the deliverable follows this sub-plane approach.

1.2 Approach and methodology

The ultimate deliverables for WP4 will be the software implementations of each component in the ARCADIAN-IoT Vertical Framework. To support the development, the agreed approach is to deliver 3 reports as progress is made which are envisaged to accompany the software implementation as follows:

Deliverable	Timeline	Description
D4.1	M12	Report: An overview of the components and their requirements, analysis of state of the art, any current resources and next steps
D4.2	M20	SW Implementation: First Prototype of vertical plane components. Report Technical design detailing implementation, deployment and full external API specification for integration in each of the domains to be carried out in WP5 and report on development status.
D4.3	M30	SW Implementation: Final Prototype of vertical plane components. Report: Final Technical design and report on final S/W release

2 IDENTITY MANAGEMENT PLANE

The research activity related to the Identity Management plane is part of of Task 4.1, which addresses four components: **Decentralized Identifiers**, **eSIM – hardware based identity and authentication**, **Biometrics** and **Authentication**.

2.1 Decentralized Identifiers

2.1.1 Overview

2.1.1.1 Description

As described in the W3C DID Core Specification [3] ***"****Decentralized identifiers (DIDs) are a new type of identifier that enables verifiable, decentralized digital identity. A DID refers to any subject (e.g., a person, organization, thing, data model, abstract entity, etc.) as determined by the controller of the DID. In contrast to typical, federated identifiers, DIDs have been designed so that they may be decoupled from centralized registries, identity providers, and certificate authorities. Specifically, while other parties might be used to help enable the discovery of information related to a DID, the design enables the controller of a DID to prove control over it without requiring permission from any other party. DIDs are URIs that associate a DID subject with a DID document allowing trustable interactions associated with that subject.".*

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

The DID is a URI composed of three parts; scheme identifier, a DID method and a specific identifier within the DID method, and resolves to DID Documents. The DID solution will follow the standard architecture model as portrayed by the following figure in the DID Core specification:

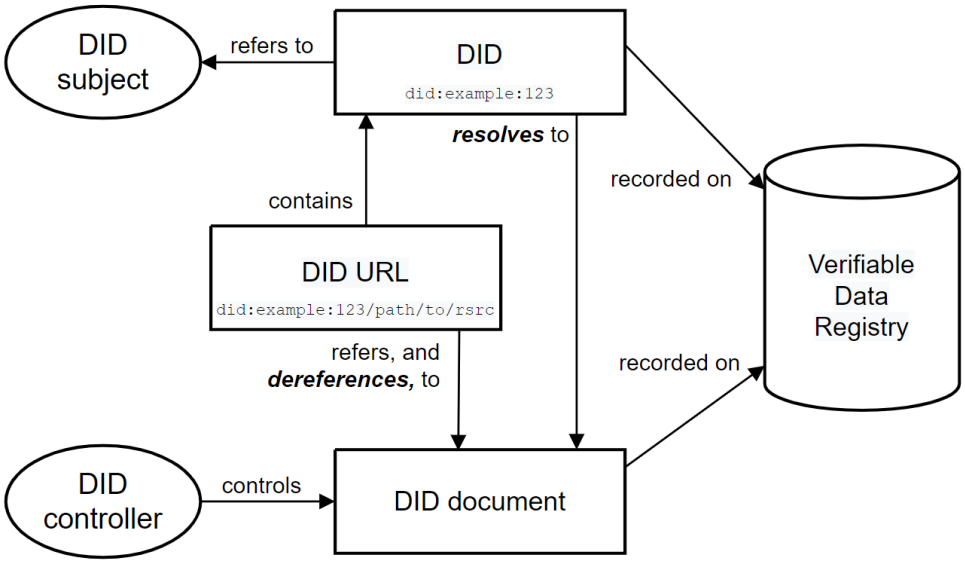

Figure 4 - ARCADIAN-IoT DID solution's basic architecture model [3]

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 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 will also be analysed for their suitability to the ARCADIAN-IoT framework and use cases, considering that latest DID methods also provide trusted DIDs 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 0.

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

2.1.1.3 Objectives and KPIs

The primary objective is to create a decentralized identity solution based on Self-Sovereign approach and therefore contribute to the overall ARCADIAN-IoT decentralized framework for IoT systems.

The Decentralized Identifiers component will be evaluated against the following KPIs:

- To support at least two of the use case domains
- To support authentication for persons and IoT devices

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 Content Addressable Storage to host the DID Docs and interact with a blockchain to provide a notarised trust anchor, as described in the Sidetree specification-[3]. That said, other DID methods will also be analysed for their suitability to the ARCADIAN-IoT framework and use cases, considering that latest DID methods also provide trusted DIDs that 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.

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 and 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 Integration with existing distributed and decentralised systems supporting DIDs

As can be seen from the following table obtained from W3C DID Specification Registries, 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 further below.

Table 1 DID Method Table

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:btcr	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

- **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 on the distributed IPFS 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, 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 users DID Doc, hecould 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 so there is no difference in the risk here. Just to 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 repository: https://github.com/decentralized-identity/Sidetree , 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 is available here https://github.com/decentralized-identity/ion-tools#ionjs, and it is seen that the native key algorithms supported are: secp256k1 and Ed25519.

- **did:ebsi [12][14]**

European Union is supporting the adoption of Self-Sovereign identity under the European Blockchain Services Infrastructure[1] (EBSI) and within that initiative the European Self-Sovereign Identity Framework[2] (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 [20] and organisations that want to test their wallets with the ecosystem.

Within ARCADIAN-IoT a sub-objective is to support eIDAS Bridge [21] within the ESSIF project where a service can issue Verifiable Credentials to a user.

An important note on the integration to support the did:ebsi 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 the cryptographic key algorithm supported by EBSI at this time is secp256k1.

As the open source SSI frameworks under consideration to support Verifiable Credentials in section 0 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

[1] https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Home
[2] https://decentralized-id.com/government/europe/eSSIF

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 [4]. 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. This means that 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.

2.1.2.2 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 Content Addressable Storage 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 Content Addressable Storage to point to Decentralized Identifiers and access the Public Keys in the hosted DID Documents. Such an implementation is depicted in Figure 1 shown below.

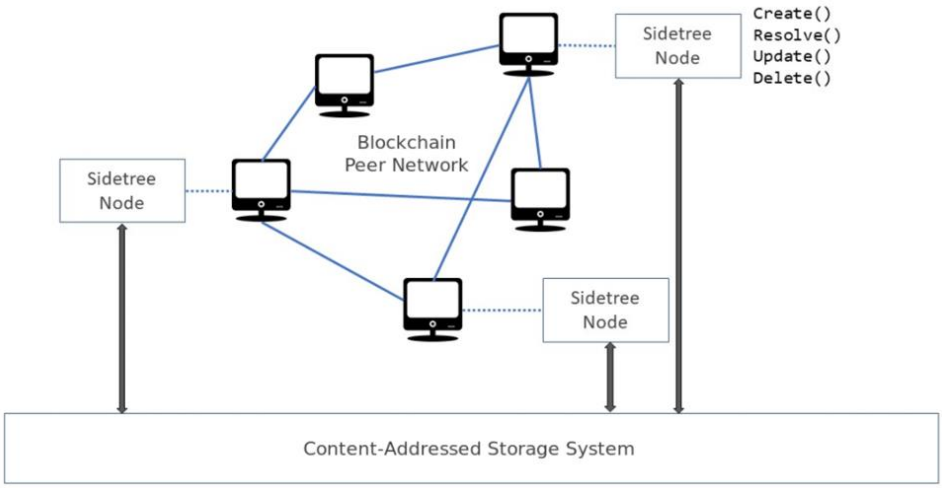

Figure 1 Sidetree DID Method Overlay network [17]

The description and Figure 2 below demonstrate in more detail the overall open source Transmute implementation of the Sidetree [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 [17] 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.

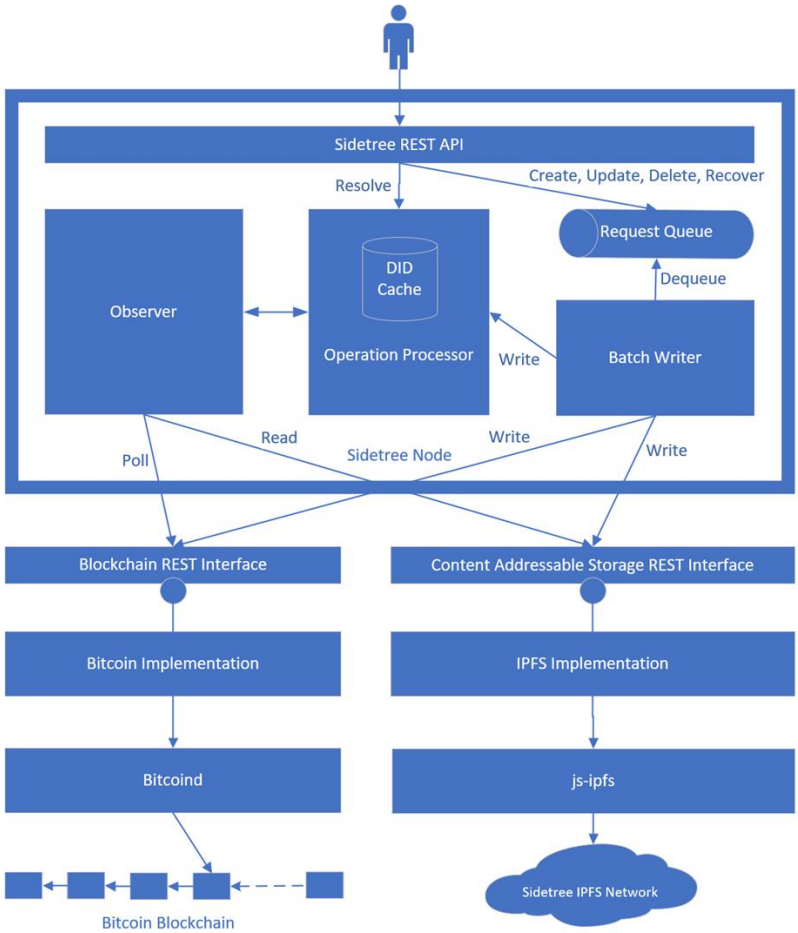

Figure 2 Sidetree implementation by Transmute Industries [16]

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 Content Addressable Storage Interface, as per Sidetree file structure specification.

Next, Sidetree batch file information will be stored into anchor index by a witness function onto the bockchain. 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 public blockchain events to identify Sidetree operations, then publishes the operations into data structures that can be used for efficient DID resolutions.

2.1.2.3 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 [18]**

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 [19]**

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 dealing in 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 0. However, it is not supported out-of-the-box with other SSI implementations that may use alternative protocols such as SIOP.

2.1.2.4 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 methods analysed in the previous sections, and therefore the SSI Framework under discussion in section in 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 [22] 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 [5], dropped the ZKP Predicates in favour to support the other privacy much sought after features of selective disclosure, non-linkability of VC signatures and credential holders, as described above.

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

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

2.1.3 Current resources

2.1.3.1 Sidetree anchored on Ethereum

Currently Atos is prototyping the Transmute industries Sidetree implementation with a private Ethereum providing the anchoring witness blockchain function based on the open source implementation [24].

Once this is successfully prototyped it needs to consider:

- The integration with the Permissioned Blockchain that is being deployed in WP3
- the key algorithm support compatibility with that used by the Self-Sovereign Identity framework that will be employed for the Verifiable Credentials.

2.1.3.2 Hyperledger Aries Self-Sovereign Identity Framework

Atos have an SSI prototype asset called LedgerUself built on top of Hyperledger Aries, as described in section 3.1. It supports the standard DIDCOMM protocol [23] for DID Exchange and presenting and issuing Verifiable Credentials.

For interwork with constrained IoT Devices that are not able to support the full SSI stack with Verifiable Credentials, the IoT device solution providers need to investigate supporting the DIDCOMM specification through available open source implementations

based on RUST such as is available at https://github.com/decentralized-identity/didcomm-messaging.

Additionally, if it is decided to use this SSI Framework it is needed to investigate adding SIOP protocol for DID Authentication to work towards interoperability with the EBSI DID specification.

2.1.4 Future work

To identify the DID Method(s) to be supported in ARCADIAN-IoT considering:

- their applicability to the use cases,
- the SSI Framework selected for the project and
- privacy preserving cryptographic signatures used in verification methods.

To continue to prototype a Sidetree based DID Method that anchors on a blockchain and considering the Permissioned Blockchain network selected in D3.1 [4]

2.2 eSIM – Hardware-based identification and authentication

2.2.1 Overview

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), integration with other relevant technologies (e.g. artificial intelligence, or reputation systems triggering actions in the devices secure element), and even in terms of 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 domain's IoT and personal devices through the provisioning the ARCADIAN-IoT eSIM profile to the eUICC (hardware in the device that receives eSIM profiles), which will also act as Root of Trust (RoT) for other components.
b. **Secure Element** (SE) capable of storing identity and authentication credentials at devices hardware level, and use them in ARCADIAN-IoT 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** with specific actions of self-protection and self-recovery according to devices threat level / trustworthiness.

eSIM will be, therefore, a relevant security agent for connected devices with several roles depicted in Figure 3 eSIM component overall view

. In this section, where the *Identity management* is targeted, the report of the ongoing eSIM-related work will focus on its role related with the item *b.* above.

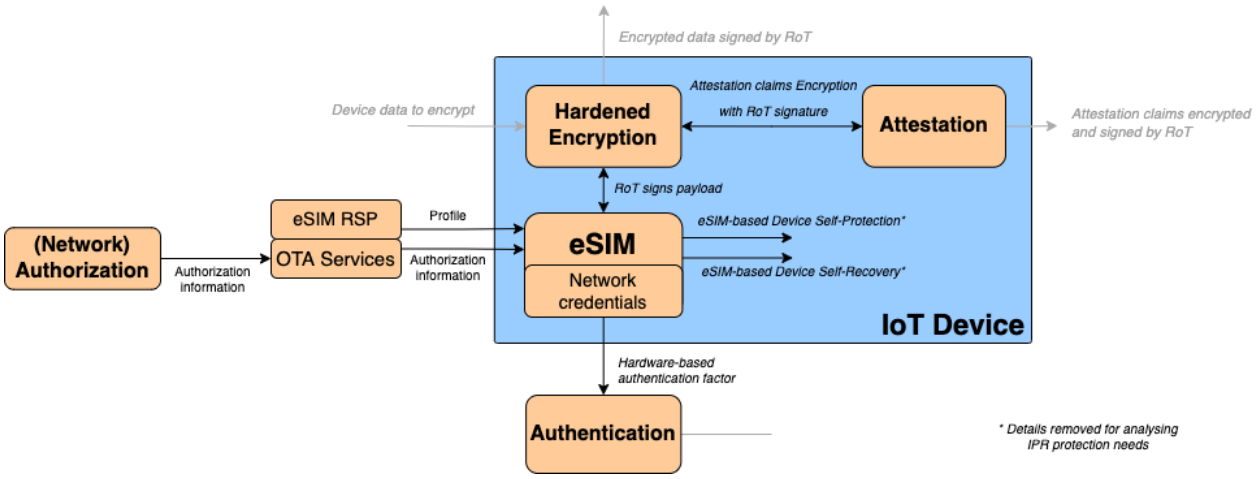

Figure 3 eSIM component overall view[3]

In that context, ARCADIAN-IoT will rely on a multi-factor authentication process to identity and authenticate users and devices. The **network-based authentication** 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 authentication, whose credentials and processes are securely stored at hardware level in the device eUICC.

2.2.1.1 Description

The network-based authentication component leverages the secure, reliable, and widely used authentication mechanism from 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 cyphering (details can be found at[4]. These well-accepted security processes rely on SIM technologies to have stored the information and processes necessary, in the device side. 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 in the IoT-related solution.

Truphone already has a patented[5] experimental proof of concept of this technology in TRL 3 and aims to bring it to TRL 6 within the context of ARCADIAN-IoT, researching to enhance it towards the demonstration in the three domains targeted in the project. Other objective is to turn this technology agnostic to the IoT devices characteristics in terms of processing power / energy autonomy (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/device autonomy).

[3] Diagram from D2.5 depicting eSIM multiple roles in ARCADIAN-IoT
[4] 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%20ciphering%20key%20(KC)
[5] https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2021224624&_cid=P10-L12DVI-41405-1

2.2.1.2 Requirements

Recalling the requirements defined in a previous stage of the project, the eSIM component has the following ones related with identity management:

- **1.2.1: IoT eSIM support**: ARCADIAN-IoT IoT devices shall support eSIM, which will act as the RoT (store identification credentials and cryptographic material for Hardened Encryption), act as security enabler and provide cellular connectivity.
- **1.2.2: Personal device eSIM support**: Compliant personal devices shall support eSIM, which will act as the RoT (store identification credentials and cryptographic material for Hardened Encryption), act as security enabler and provide cellular connectivity.
- **1.2.3: Receive eSIM profile requests**: The eSIM ecosystem shall be able to receive requests for *ARCADIAN-IoT eSIM profiles* from compliant ARCADIAN-IoT devices or services.
- **1.2.4: Generate an ARCADIAN-IoT eSIM profile**: A novel and specific ARCADIAN-IoT eSIM profile needs to be developed. It will contain identity and authentication elements (related with persons and devices) and cryptographic material and processes for implementing Hardened Encryption in devices. It will also have methods for securely accessing that RoT information from the device. This profile also needs to be ready to provide connectivity.
- **1.2.5: Provision profiles to authorized devices**: An *ARCADIAN-IoT eSIM profile* shall be provisioned, over-the-air, to authorized devices, when requested by the device or by an authorized entity.
- **1.2.6: Update eSIM profile**: It shall be possible to securely update an eSIM profile over-the-air.
- **1.2.7: Access eSIM profile information**: Information in ARCADIAN-IoT eSIM profile needs to be securely accessible from the device where it is hosted.

While these requirements are (continue to be) valid to the overall eSIM component considering all its roles, and naturally including the identification and authentication in third party services with network credentials, please consider the section 2.4 as well (Authentication component) for understanding the complete set of related requirements. The requirements described in that section are not replicated here to avoid content duplication.

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:

- **Objective 2**: Enable security and trust in the management of objects' identification[6]
  - **KPI 1**. Support, at least, two identity approaches at hardware level (eSIM and CryptoChips).
  - **KPI 3.** Support, at least two robust identity mechanisms for devices and apps/services.
- **Objective 3**: Enable distributed security and trust in management of persons' identification
  - **KPI 4**. Enable, at least 3 multiple simultaneous identification approaches for

[6] The KPI 2 of objective 2, "Avoid single trusted entities through decentralized approaches (eSIM identity approach)", by mistake associated eSIM with decentralized approaches. As clearly shown in the objective description of the grant agreement the decentralized approaches refer to "Decentralized Identifiers in combination with Verifiable Credentials", not to eSIM.

persons.

2.2.2 Technology research

The current approach to the network-based authentication component is pictured in Figure 4. Basic assumptions necessary for the technology research are (1) 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 compliant third-party. The device communicates with the *Notarizer* (Figure 4), subcomponent that is positioned in the core network, sending it the message to deliver along with its destination. The *Notarizer* uses the device's network identifiers to confirm its identity in a subscriber database at core network and, if the validation is successful, this means that the device is known by the operator, is a valid subscriber of the network and, obviously, has already successfully authenticated and attached to the network. 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 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 an *Network authenticator* positioned in ARCADIAN-IoT framework. This authenticator, validates the token and sends the result to ARCADIAN-IoT *multi-factor authentication*, to join the result with the one of other simultaneous authentication factors (further details in the authentication section - 2.4).

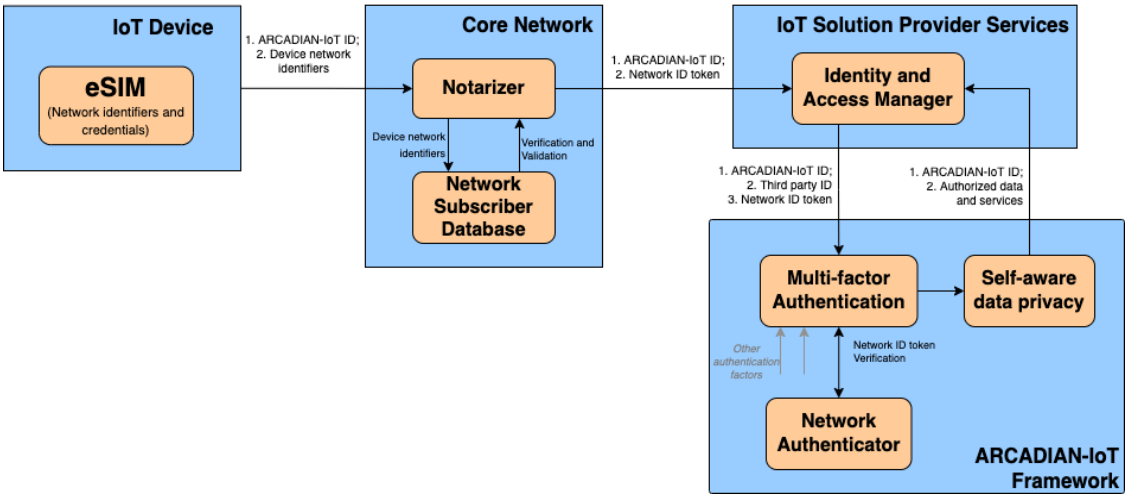

Figure 4 - Architecture of the Network-based authentication in third-party services

This token-based authentication is based on Open ID Connect[7] (Figure 5). 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.

[7] Final: OpenID Connect Core 1.0 incorporating errata set 1

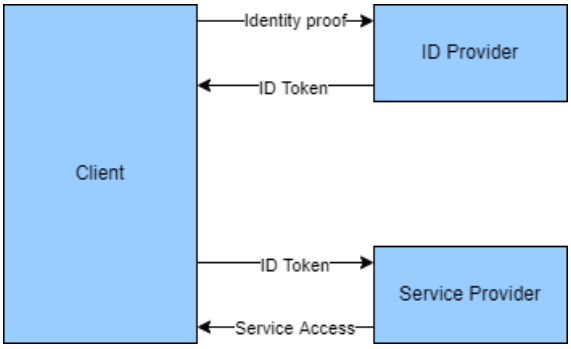

Figure 5 - Open ID connect architecture

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. ID tokens usually contain a URL that allows the Service Provider to confirm the validity of the token.

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.

2.2.2.1 Technical analysis

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 Truphone, which has a proof of concept, considered to be in TRL 3. With the research done in ARCADIAN-IoT 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.

Table 2 depicts the current prototyping efforts done in the context of ARCADIAN-IoT.

Table 2 – Current status of the network-based identification and authentication component

#	Subcomponent	Brief description	Prototyping status
1	**Notarizer**	Validates, at the network core, an IoT device as a known network subscriber with successful authentication, and provides an ID token for authentication in compliant third-party services	First prototype done in March-22. The current prototype has the functionality described in Figure 4 but performs a different flow – the network ID token is returned to the device who connects directly to the third-party service (the exact flow of OpenID Connect). Unit and integration testing with a real device with an eSIM ongoing.
2	**Network-based authenticator**	Confirms the validity of the provided ID token	Prototype with AWS IoT Core custom authenticator (and related lambda function) done in March-22. Unit and integration testing with a real device ongoing.

The above-mentioned subcomponent prototypes were developed using standard technologies (e.g. JWT for the ID token, and Rust and GO for the *Notarizer*). Further studies regarding the specific technologies used in the current implementation, as well as the current status of the research in terms of constrained IoT devices is expected to be published in 2022.

2.2.3 Current resources

The patent with the previous work can be found at: https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2021224624&_cid=P10-L12DVI-41405-1.

2.2.4 Future work

As future work, for the next reporting period, we envision the following:

- Demo showing the current subcomponent prototypes.
- First assessment of the technology.
- Publication of the research on the agnostic solution for the IoT devices, considering constrained equipment, in a peer reviewed conference or well-accepted journal.
- Delivery of a second prototype with the enhancement of the technology according to the assessment made, IoT domains knowledge, and the integration within ARCADIAN-IoT framework needs, namely in the multi-factor Authentication component.

2.3 Biometrics

2.3.1 Overview

2.3.1.1 Description

The biometrics component adds another factor to identify persons entities, relying on their biometrics such as face characteristics. This component will support facial recognition considering operational challenges. The UWS Biometric component will be responsible for receiving a set of photos from the client with the face that constitutes the DB to perform face identification. Besides, the component will process the video feed received by the drone. Internally, first the component will run a face detection algorithm to search for a face, this algorithm is Convolutional Neural Network-based and will locate and crop the face that appears in the video. Immediately after, the face identification algorithm will compare the face extracted against ones available in the database. In addition, we will explore the possibility to provide together with identification result, the bounding box that represents the part of the scene where the face has been identified to allow a GUI to create an overlay figure. The Figure 6 shows the main interfaces foreseen for this component. Note that for brevity ARCADIAN-IoT is referred to as AIoT in the figure.

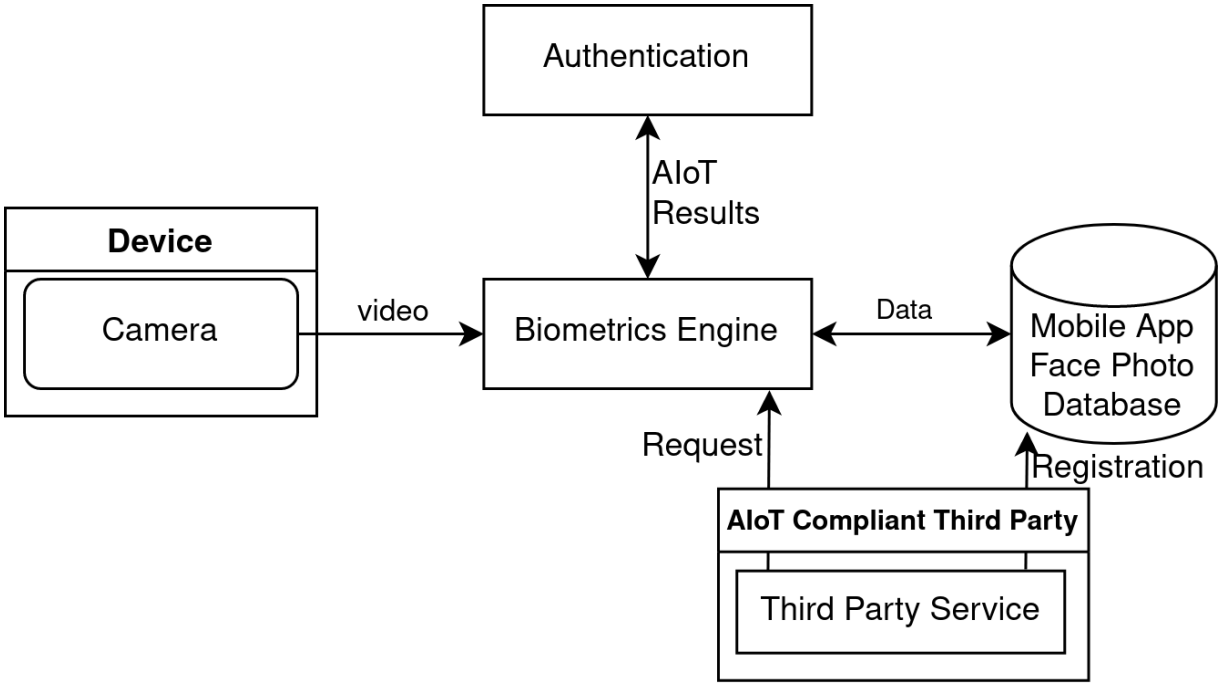

Figure 6 Biometric component interfaces

2.3.1.2 Requirements

The following requirements has been identified for the Biometrics component:

- The system requires several photos of the client's face to perform face identification against a video feed.
- The algorithm requires the reception of a video feed coming from the drone in order to allow the biometric algorithm to perform the face identification.
- The algorithm requires HD video resolution.
- The system requires GPU support to execute the algorithm efficiently.
- The algorithm requires adequate lighting to perform accurately.
- The system requires connectivity to a Key Management Service to decrypt the video encrypted by the drone (if the video feed is encrypted).
- The system requires direct communication with the Authentication Component to provide them with the required biometrics results.

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 identification of the clients.
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:

1) Number of frames that can be analysed per second (KPI: 10 Frames per Second)
2) End-to-End Delay of the whole biometric process. (KPI: 2000 milliseconds)
3) Accuracy achieved by the face identification algorithm (KPI: above 90% F1-Score or similar accuracy metrics).

2.3.2 Technology research

Biometric identification has been widely applied to myriad of applications. In ARCADIAN-IoT project, the focus will be on applying biometrics to 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 condition between camera and individual) while considering necessary privacy preservation.

This drone-based Biometric component will recognise a person (focusing on ARCANDIAN-IoT users such as the drone pilots for authorisation and the user of the Drone Guard Angel use case service) through analysing his/her 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.

1) Face database

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:

Database	Size (images)	Identities	Drone Friendly	Notes
DroneSURF	441,000	58 people	Yes	Missing different lightning conditions.
LFW	13,233	5749 people	No	Close range images.
Color FERET	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 execution, 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 a 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 20 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 further the camera is.

Algorithm	Speed (ms)	Accuracy	Far Distance Performance

ArcFace	50	99.84%	No results reported
VGG-Face	110	98.95%	Low performance.
Dlib	10	99.38%	Regular performance.

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 executing the algorithm in real-time. In the following table, four most common platforms are explored and compared based on four different factors:

Platform	Execution Environment	Code	Deployment	Compatibility (OS)
TensorFlow	CPU / GPU	Open Source	Medium	High
PyTorch	CPU / GPU	Open Source	Medium	Medium
OpenCV	CPU	Open Source	Easy	Medium
SNPE	Snapdragon	Proprietary	Medium	Low

Table 5 Machine Learning frameworks

2.3.2.1 Technical analysis

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. The main solutions to complete this dataset are to manually record and label new videos and to include and mix available public datasets. 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.

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.

Finally, 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.

2.3.3 Current resources

As stated in the ARCADIAN-IoT project proposal, UWS is interested in exploitation of the algorithms, platforms and applications developed as a result of our partnership in the ARCADIAN-IoT project. Through the UWS research and enterprises services department and in collaboration with other partners, we are willing to explore the commercial opportunities of our research

outcomes in IoT, 5G and beyond networks. This approach is aligned with our business model and with the UWS intellectual property policies.

For this reason, UWS has not the initial intention to make any release of the source code for this component and thus it will not be made available in any public or private repository or server outside of our premises. For integration purposes, a functional prototype of the biometrics component will be released for the ARCADIAN-IoT consortium.

2.3.4 Future work

Technical analysis section has highlighted where the main efforts are needed in order to have a successful biometrics system. First, DroneSurf dataset should be completed by adding new images of people in different lightning conditions until the system achieves more accuracy than current SOTA works. Moreover, most of the efforts will be focused on the implementation of a face identification algorithm able to identify people in far distances as already mentioned in SOTA studies. The algorithm should also be implemented in a GPU compatible execution platform such us TensorFlow or PyTorch. The future steps are then as follows:

- Processing of a face DB suitable to fulfil the aims of Arcadian project Domain A (Drone Angel), and annotation of the face DB.
- The iterative process of designing and training of the AI algorithms with the face DB in order to achieve a suitable algorithm.
- Design of the system and its interfaces (including interfaces with Authentication system, Drone video feed and key management server),
- Integration of the algorithms with the system, testing and validation, demonstration, and project dissemination including publication of the results and further exploitation.

As depicted in figure 7, the development of this component has been divided in different tasks or sub-tasks to be accomplished in the task 4.1 of the project. The duration of each sub-task and the workload of each stage has been determined to meet the expected targets and deadlines for Work Package 4 (see Gannt diagram in Figure 7).

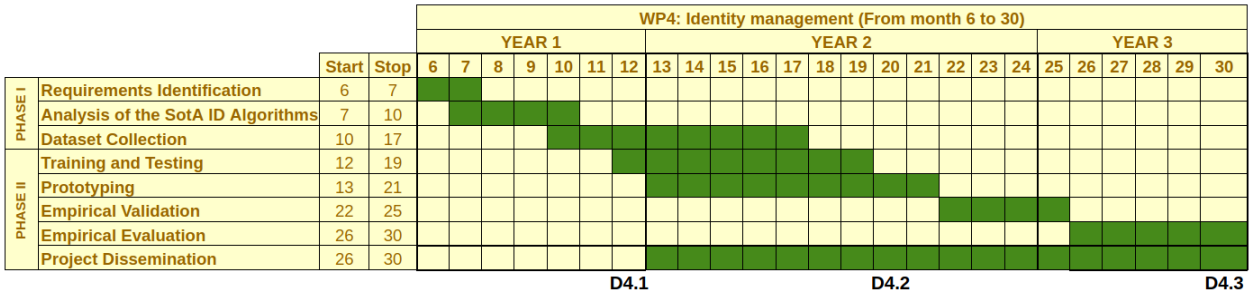

Figure 7 Gantt diagram Biometrics Component in Work Package 4.

The first phase of the project (Phase I) focused more on the theoretical aspects of the project such as requirements analysis and design, which has been successfully accomplished in the first 6 months of the task 4.1. Currently, the UWS team is engaged in the collection of a comprehensive dataset of faces from different distances.

While collecting the dataset, the UWS team has also started the training and validation of this preliminary dataset. This a labour intensive and highly computational task due to the large number of images and the complexity of the algorithms. The dataset will evolve over the time, including new data and removing biased one, therefore, this work is a task strictly related to the algorithm itself. After completing this process, the UWS team will focus on its first functional prototype. A first version of this prototype will be available during the second year of the project.

After prototyping and as the last step in the development of the component, the prototype will be empirically validated and evaluated. Hence, a final version of the prototype will be provided in deliverable 4.2 and an empirical validation and evaluation of the overall performance of this component will be reported in deliverable 4.3. The project dissemination including publication of the results, and further exploitations will be carried out during year 2 and 3.

2.4 Authentication

2.4.1 Overview

2.4.1.1 Description

ARCADIAN-IoT authentication processes are widely focused and described in the previously defined use cases for the three IoT domains targeted on the project. Considering those use cases and the project objectives, ARCADIAN-IoT authentication will rely on a multi-factor authentication process to identity and authenticate persons and devices in IoT solution providers. The targeted input factors are other components from the framework, specifically:

a. Decentralized identifiers and credentials (described in section 2.1);

b. eSIM hardware-based/network-based identification and credentials (described in section 2.2); and

c. Biometrics data (described in section 2.3).

Figure 8 depicts the main objective of ARCADIAN-IoT authentication component: to be the aggregator of the several authentication factors, to provide a robust authentication mechanism for the mentioned entities (persons and IoT objects) in ARCADIAN-IoT third party services. The same figure also elucidates an aspect, that was in fact a research result from the current reporting period, which is that the outcomes from the component will feed the Behaviour Monitoring component and the self-aware data privacy component.

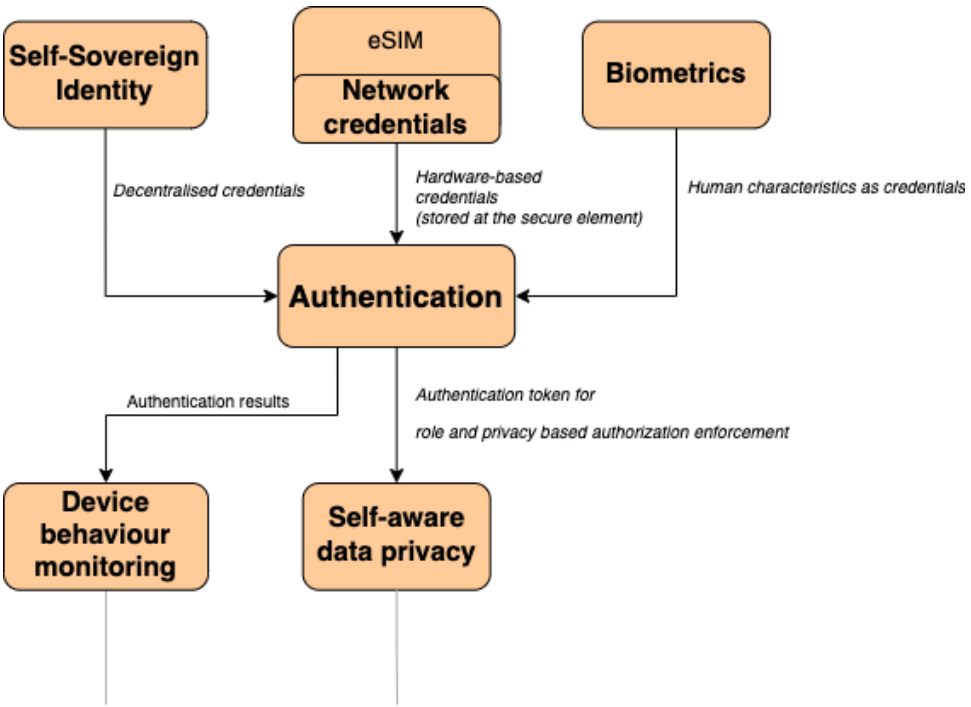

Figure 8 - ARCADIAN-IoT authentication high-level architecture

The relation with the Behaviour Monitoring component is for allowing to understand authentication events that may relate with a threat. The relation with the self-aware data privacy is for, after a successful authentication, performing identity and access management control according to the user defined privacy related preferences (for himself/herself or for his/her devices), and for role base access control (to which services a person or device can access).

2.4.1.2 Requirements

Three requirements were previously defined for the authentication component:

- **1.4.1: Authenticate persons in ARCADIAN-IoT services**: Persons should be able to be identified and authenticate in compliant ARCADIAN-IoT services using SSI (DID, VC), network credentials from their personal devices and/or with their biometric characteristics. Some of this information is stored in the personal device RoT and other uses a decentralized approach.
- **1.4.2: Authenticate devices in ARCADIAN-IoT services**: Devices (IoT devices and personal devices) should be able to be identified and authenticate in compliant ARCADIAN-IoT services using SSI (DID, VC) and network credentials. Some of this information is stored in the device RoT and other uses a decentralized approach.
- **1.4.3: Identify and authenticate apps and services in ARCADIAN-IoT**: Compliant apps and services should be able to be identified and authenticate in ARCADIAN-IoT framework with two robust identity mechanisms.

According to the research made so far, requirements 1.4.1 and 1.4.2 continue to be valid and feasible. The requirement 1.4.3, in its detail, mentions the use of two identity mechanisms for apps and services. The mechanisms foreseen in the grant agreement, decentralized identifiers, biometrics and the network credentials stored at the eSIM don't allow to fulfil this requirement. While it is possible to identify services with decentralized identifiers, biometrics are not used for such a purpose, neither is eSIM, hardware applicable to IoT and mobile devices, but not to services stored, e.g., in Cloud servers. Therefore, at the moment, the consortium just identified one identification mechanism for apps and services. Further analysis and decision regarding this topic will be made in the forthcoming reporting period.

2.4.1.3 Objectives and KPIs

ARCADIAN-IoT authentication component is expected to contribute to the accomplishment of the following objectives and KPIs:

- **Objective 1**: To create a decentralized framework for IoT systems - ARCADIAN-IoT framework
- **Objective 2**: Enable security and trust in the management of objects' identification
  - **KPI 1**. Support, at least, two identity approaches at hardware level (eSIM and CryptoChips).
  - **KPI 2**. Avoid single trusted entities through decentralized approaches
  - **KPI 3.** Support, at least two robust identity mechanisms for devices and apps/services.
- **Objective 3**: Enable distributed security and trust in management of persons' identification
  - **KPI 4**. Enable, at least 3 multiple simultaneous identification approaches for persons.

2.4.2 Technology research

ARCADIAN-IoT authentication relies on multiple technologies, from multiple partners, to accomplish its objective. This fact influenced the research strategy, which had the objective of settling a well-accepted vision for the component among all the partners involved. In the current reporting period, the research departed from the use cases and related requirements previously defined to, with that knowledge as base, plan a set of focus group sessions with the partners responsible for each authentication factor, and with the partner responsible for the self-aware data privacy component. The main result of the research, at the moment, is the detailed architecture presented in

Figure 9, well-accepted among the partners.

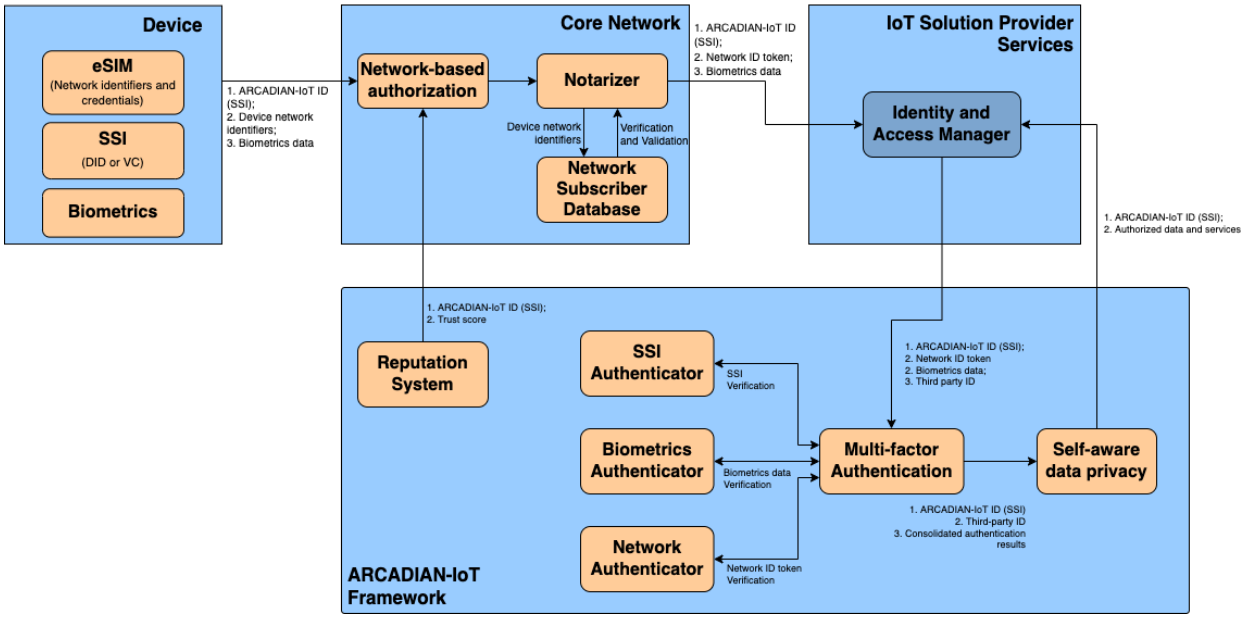

Figure 9 - Architecture from ARCADIAN-IoT multi-factor authentication

The presented architecture depicts the following process:

1. Authentication starts at the device (IoT device or personal device). For the analysis simplicity we'll consider just the personal device case, which uses the three authentication factors (IoT devices just use the eSIM and the decentralized approach; don't use the biometrics factor – being therefore a subcase of the one being presented). The process initiates with the device requesting authentication by sending its network identifiers, SSI claims and the biometrics data, to the intended IoT Solution Provider Services. The involved partners agreed that the three factors will be used to strengthen the identification and authentication process, but, after authentication, the ID that will circulate among the technical components to univocally identify an entity (person or device) will be its SSI data.
2. The sent data, before arriving the internet, passes through the cellular network core, where the network-based authorization component assesses the device trustworthiness (provided by the reputation system) and applies the related policies. If the device is allowed to continue, still in the network core, the device network identifiers are verified, to assess if the device is a known subscriber (successfully authenticated in the network according to GSM standards). If so, a *Network ID token* is generated and attached to the message, certifying that the subscriber is known by the operator.
3. The authentication request follows to the intended IoT Solution Provider Services, now comprising: the Network ID token, the SSI claims and the biometrics data. The third-party service joins its own ID (third-party ID) and requests the authentication from ARCADIAN-

IoT multi-factor authentication, expecting to receive back which are the services and the data that that entity is entitled to.

4. After receiving the request, the Multi-factor Authentication component directs each of the identification factors data to its correspondent authenticator, joining for the identification verification the ARCADIAN-IoT ID and the third-party ID.
5. The SSI authenticator verifies the SSI claim validity, returning the result to the Multi-factor Authenticator.
6. Having the ARCADIAN-IoT ID, the third-party ID and the Network ID token, the Network-based authenticator verifies its validity, meaning, if the Network ID token – representing the network subscriber ID - matches the one from the ARCADIAN-IoT ID, informing the Multi-factor Authenticator component.
7. Simultaneously and likewise, having the ARCADIAN-IoT ID, the third-party ID and the biometrics data, the biometrics authenticator verifies its validity, meaning, if the biometrics data provided matches the biometrics data from the ARCADIAN-IoT ID, informing the Multi-factor Authenticator of the result.
8. The Multi-factor Authenticator receives the results from the assessments above and, if the authentication was successful in the three, informs the self-aware data privacy component that the given ARCADIAN-IoT ID had a successfully authentication. At this moment, the three authentication factors were already verified, and the ID is confirmed.
9. The self-aware data privacy component will now generate the information about the authorization that that entity has in the requesting third-party, in terms of services (according to the roles) and data that it can access (according to user defined privacy rules). This information is returned to the requesting third-party.
10. According to the authorization information provided, the third-party grants the access to the data and services.

The vision depicted above is expected to be enhanced in an agile research approach, considering the new knowledge built, according to what's defined in the future work.

2.4.2.1 Technical analysis

The authentication component is, in itself, an aggregator of other components results. It is expected that the complexity of research done within the scope of this component will be towards the articulation of the authentication factors and not in a highly complex individual component.

2.4.3 Current resources

N/A

2.4.4 Future work

As future work, for the next phase, we intent to schedule another set of work sessions among the involved partners to define the technical integration details among all the components. Considering the agile methodology (or action-research methodology), any adjustment of the current results (the presented architecture) that outcomes from the analysis of the technical research results will be welcome, because it will ensure an enhanced end result. To ensure alignment between the several technologies, within the scope of work of this component will be ensured that every partner is involved in every change decision. In the next phase it is expected to have a first prototype of all the involved components, ready for a first integration and integration testing, which will feed the research of the forthcoming period and the work done in WP5. Within the next phase, domain owners (owners of the third-party services mentioned above) will also be

involved in the technical meetings.

Furthermore, the following topics will as well be targeted in the research of the next reporting period:

- Person and device registration in ARCADIAN-IoT.
- Re-authentication processes: security rules (time-based; event-based) that define when a device or a person needs to authenticate again to access third party services.
- Apps/services robust identification factors.

3 TRUST MANAGEMENT PLANE

The research activity related to the Trust Management plane is part of of Task 4.2, which addresses four components: **Verifiable Credentials**, **Network-based Authorization**, **Reputation System** and **Remote Attestation.**

3.1 Verifiable Credentials

3.1.1 Overview

3.1.1.1 Description

ARCADIAN-IoT will provide an identity management solution that is built on W3C Verifiable Credentials specification [6] that is a core standard that is helping to standardise the Self-Sovereign Identity (SSI) approach in decentralising identity management. 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 also 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
- to further enhance trust through the use of VCs, by enhancing implementations towards standard’s interoperability.
- use Verifiable Credentials in combination with Decentralized Identifiers (DIDs), based 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 [21] within the ESSIF project where a service can issue Verifiable Credentials to a user.

KPIs defined for Verifiable Credentials are listed below:

- Support at least one domain use case with Verifiable Credentials

3.1.2 Technology research

3.1.2.1 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 [25]** is an evolution of the uPort open source SSI software that was one of the first pioneers of SSI from 2015. uPort's technical architecture and open source libraries. started to manifest limitations due to changes in maturing standards that meant many changes throughout the code base and also its tight integration with on-chain identities. An evolution to a new open source framework has resulted in a new modular architecture based around a library of core functionality, which allows the developer community to easily interface with and extend its functionality as needed, for example with additional DID methods, key management, protocols.
- **Veres One [26]** 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 [27]** 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 [28]** is an open source evolution from Hyperledger Indy that cteates 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 [29]** 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 [30]** 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 [31]** 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 [32]** is an open source and standards based Rust implementation of decentralized digital identity, also known as Self-Sovereign Identity (SSI). 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 [33]** 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 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.

3.1.2.2 Interoperability

Due to initial SSI developments preceding much of the standards work and differing rival technologies each implementation was never going to be able to interwork with any 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 [34] 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
Communcation/Transport	DIDComm V2 + transports	OIDC + SIOP	DIDComm V2 + transports	DIDComm V1 + DIDComm V2 Envelope + transports	DIDCOM V1 + transports

Figure 10 Protocol support for SSI [34]

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 11 Crytpgraphic technology [34]

Hyperledger Aries, MATTR, Spruce and Veramo are amongst the active participants in this Interoperability WG.

Interoperability testing is also supported by W3C with issuing and verification of VCs for testing

available here[8].

- **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) [37] and OIDC-4-Verifiable-Presentations (OIDC4VP) [36] build upon the well establish 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 [38] 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.

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 [35] 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 are promoting DIDCOMM.

3.1.2.3 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 also only integrated with IOTA Tangle DID to date 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 also support a comprehensive open source solution and is active in interoperability efforts as can be seen here [39] and their work 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

[8] https://github.com/w3c-ccg/vc-api

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 here[9]. It is amongst these characteristics that Atos Research and Innovation chose Hyperledger Aries GO to build its Self-Sovereign Identity solution called Ledger uSelf, which is described in some more detail in the following section.

3.1.3 Current resources

3.1.3.1 Ledger uSelf

Atos is building the Ledger uSelf solution in order to simplify the adoption of Self-Sovereign Identity by creating a broker that acts as a wrapper on top of the Aries Agent, which in turn makes integration easier for Relying Parties. 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)[10] 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.

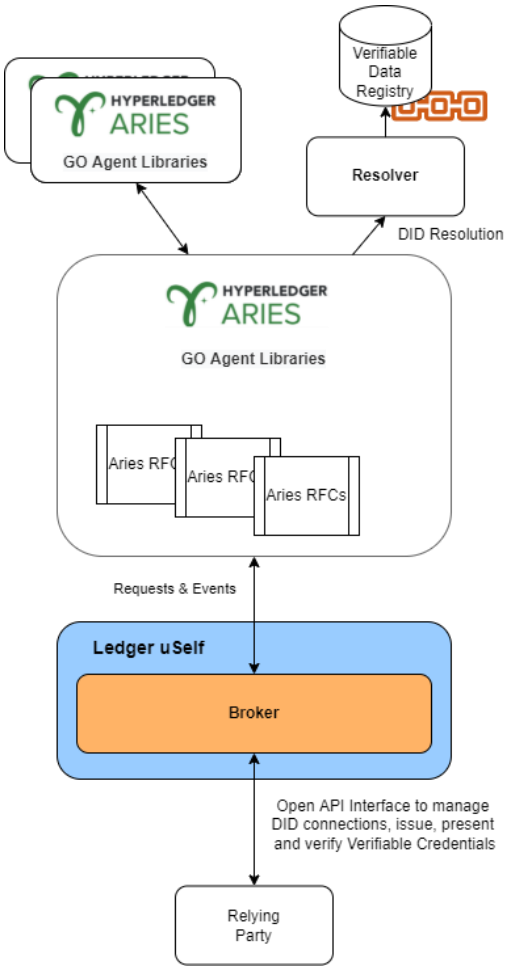

Figure 12 Ledger uSelf built on top of Hyperledger Aries GO Agent

[9] https://github.com/hyperledger/aries-framework-go/releases
[10] https://github.com/hyperledger/aries-rfcs

The following figure shows more the areas where add-on functions are provided by the Ledger uSelf Broker for simplifying and extending Hyperledger Aries capabilities.

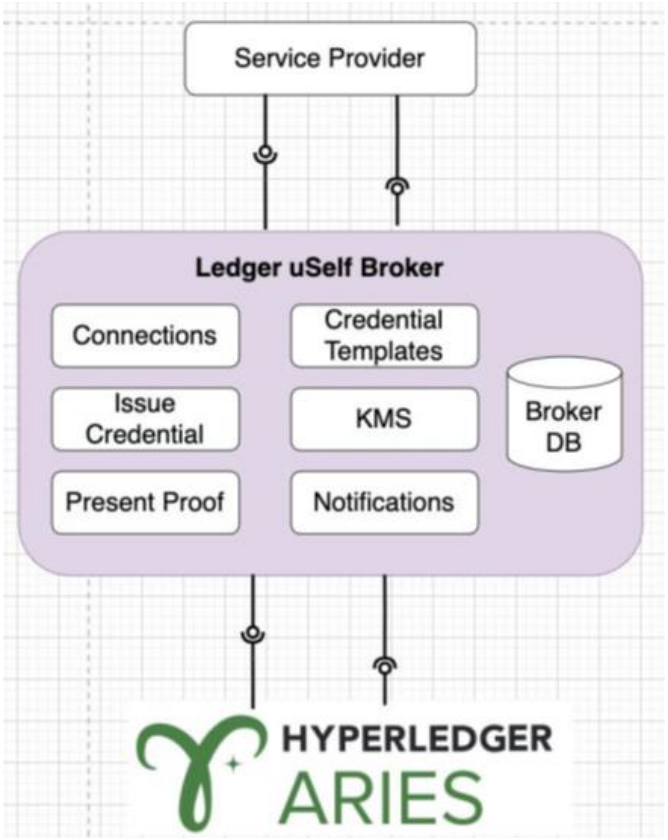

Figure 13 Ledger uSelf Broker functions

The Ledger uSelf solution also supports a mobile SSI Wallet and mediator for handling the SSI messaging to mobile SSI wallets when they are off-line.

3.1.4 Future work

The full analysis of support of Verifiable Credentials according to ARCADIAN-IoT architecture and domain use cases needs to be finalised and determine the base implementation and desirable features. Considering the ARCADIAN-IoT use cases and project objectives it is seen that the SSI features that should be considered for implementation in the next steps are as described below.

Issue, present and validate identity Verifiable Credentials for persons and things

To support things, it should be investigated: 1) Running Hyperledger Aries on MCU devices in GO and 2) Integrating another Rust based SSI Framework into the Ledger uSelf solution such as SpruceID as discussed previously.

Note: The support for SSI in the domains is currently as follows:

- The Domain A Drone use case is considering the requirements to be capable of supporting the Ledger uSelf solution.
- The Domain B use case is currently only considering the support of DIDCOMM due to

its support in IOTA and to support DID Authentication of the IoT Devices. It is considering a 3rd party OS RUST software.
- The Domain C Use case will be supported by the SSI Wallet, but the Bluetooth wearable devices will not integrate with any SSI functions directly.

Relying Party support through easy integration based on OIDC

Currently the Ledger uSelf solution does not support direct integration with OIDC. Therefore to support this in ARCADIAN-IoT it should be considered to support either:

1. Full SIOP and OIDC4VP stack to support user presentation of Verifiable Credentials
2. Integration with an OIDC Authorisation Server which integrates with Ledger uSelf and also the Relying Parties piloting in ARCADIAN-IoT

Support Verifiable Claims with Zero-Knowledge-Privacy features

As discussed above it is very desirable to support BBS+ signatures as the default cryptographic key algorithm that's supports privacy preserving features such as selective disclosure and this will be researched further with support from XLAB.

The latest GO framework code release for signing and verification algorithms is available here: https://github.com/hyperledger/aries-framework-go/tree/b1b076db898fe8c922c6dc093d3fa52d448f0c30/pkg/doc/signature

Integrate with eIDAS Bridge for issuing trusted VCs

As discussed previously, this is a desirable objective for the project and its integration will be fully considered.

3.2 Authorization: Network-based authorization enforcement and authorization distribution

3.2.1 Overview

3.2.1.1 Description

ARCADIAN-IoT has different technologies that relate with authorization, namely network-based authorization enforcement, the authorization distribution to IoT devices, and the self-aware data privacy, where users define authorization rules to the access to their data. In this section, we will focus on the network-based authorization that, in one hand, enforces trust-based authorization rules in the network, and in the other, informs devices' secure element of the trustworthiness of the device where they are (distributes authorization information to devices).

Figure 14, depicts the network-based high-level functional architecture. Specifically, in ARCADIAN-IoT, we will leverage network-based policy enforcement tools, to enable novel processes of dynamic authorization throughout ARCADIAN-IoT ecosystems. In these processes, authorization is expected to be enforced according to the entities' trustworthiness (provided by ARCADIAN-IoT reputation system), which allows to automatically act in the presence of threats, by blocking accesses from/to the network (e.g., blocking unauthorized accesses to sensitive data or unauthorized control of devices or services behaviour).

Furthermore, this network-based authorization component will also receive self-recovery routes (from the self-recovery component) to allow the automatized recovery of compromised devices

when needed.

Lastly, with the devices' trustworthiness information, the component will securely inform devices secure element (eSIM) regarding the devices level of compromise. This will allow this independent hardware that lives inside devices to take actions of protection or recovery according to devices trust level.

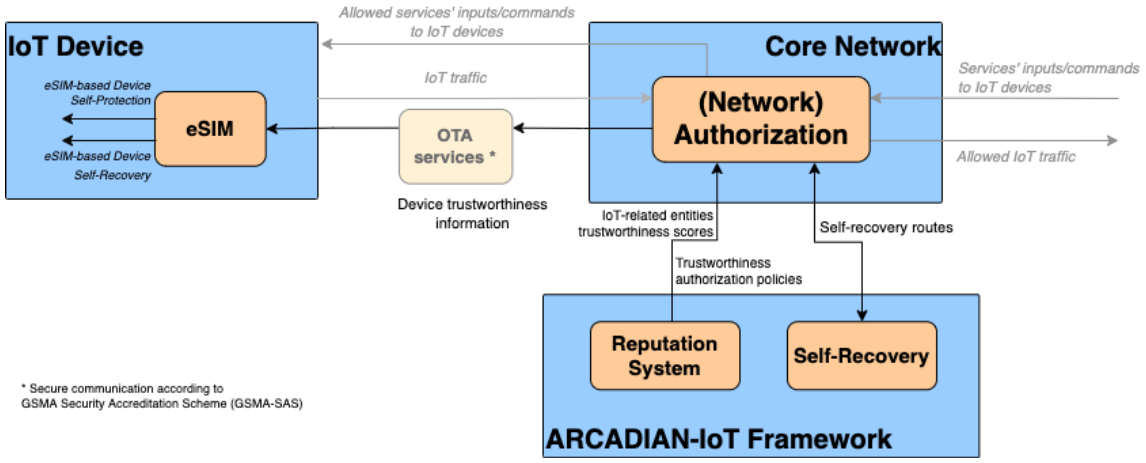

Figure 14 - ARCADIAN-IoT Network-based authorization high-level architecture

Apart from the novel authorization mechanisms described on this section, as previously mentioned, ARCADIAN-IoT framework also considers the self-aware data privacy component (as described in deliverable D3.1 [5]). An overview over the integration of these components is presented at section 2.4 (Authentication).

3.2.1.2 Requirements

Previously in the project were defined two requirements for the network-based authorization component [1]:

- **5.2.1: Dynamic network-based authorization enforcement:** A network-based enforcement tool (placed in the core network) will control devices, persons, and services access (I/O) to Internet resources based on those entities' reputation/trustfulness, ensuring, for example, that compromised devices just access (or are accessed by) recovery services.
- **5.2.2: Dynamic RoT/eSIM authorization enforcement**: ARCADIAN-IoT eSIM profile shall have methods for enforcing security authorization.

These requirements continue to be valid and feasible. Details on the requirement 5.2.2 were purposely removed for future IPR protection.

3.2.1.3 Objectives and KPIs

Although the grant agreement does not explicitly state any objective or KPI for the authorization component, this technology implicitly contributes to:

- **Objective 1**: To create a decentralized framework for IoT systems - ARCADIAN-IoT framework
- **Objective 4:** Provide distributed and autonomous models for trust, security and privacy – enablers of a Chain of Trust (CoT)
- **Objective 6**: Self and coordinated healing with reduced human intervention

- **KPI 1.** Recovery, at least 95% of the system functionalities prior to anomalous behaviour.
- **KPI 2.** Support coordination of recovery to pre-defined trust levels.
- **KPI 3.** Reduce human intervention to the strictly required, in healing and recovery procedures.

The contribution to objective 1 is through the impact in the following objectives. The contribution to objective 4 is on the enforcement of the defined model for trust, security and privacy, being this component a relevant autonomous agent able of receiving inputs from the Chain of Trust to enforce security actions. Contribution to objective 6 is by informing the eSIM of device's recovery status, triggering the subsequent eSIM-based recovery actions.

3.2.2 Technology research

The research made in the current reporting period had three objectives: (1) define a unified vision with the partner responsible for ARCADIAN-IoT reputation system, critical component 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; and (3) have a first working prototype of the component with basic rules being enforced automatically.

In what concerns (1) and depicting the functional understanding of the network-based authorization component, it leverages the trustworthiness (i.e., their reputation, in the context of ARCADIAN-IoT) of the devices communicating into the network, and network services communicating with devices, to enforce authorization rules. The simplest high-level example of this process is the one of a device that, because it has a high trust reputation has access to all the systems and data it needs, while one with low trust reputation has communication restrictions (exact restrictions are pending the trust policies to be developed in the context of the Reputation System). A more concrete example of the full functional flow that shows the integration of the network-based authorization in a ARCADIAN-IoT scenario is the following:

> *At a given moment, and for no expected reason, drones A, B and C, all from the same brand and model, which were just turned on to be available to provide Drone Guardian Angel (DGA) service, start sending an unusual and very high amount of data (e.g. high resolution live and continuous video of its surroundings) to DGA backend services. These drones are also sending the data, decrypted, to an unknown internet service. If more drones (e.g. from the same brand and model) have the same behaviour, this will 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 security or privacy breach and is a violation of ARCADIAN-IoT compliance. These behaviours hamper the IoT devices', drones in the case, ability to perform their service (due to the suspicious behaviour and to the very high battery consumption).*
>
> *ARCADIAN-IoT Behaviour Monitoring and Flow Monitoring detect these suspicious behaviours and, while triggering protection and mitigation measures, inform the reputation system of these events. Considering the serious threats posed by the drones, the reputation system reduces their trustworthiness to the lowest rating possible.*
>
> *At that moment, automatically, the **network-based authorization** is informed of the devices 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 sending 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 the normal communication again, informing as well the devices' secure element that the device is trustworthy again.*

The aforementioned scenario allows to understand the expected actions and interactions of the network-based authorization component within ARCADIAN-IoT framework, when applied in a specific domain.

As first technical hypotheses assumed by the involved parties (TRU for the network-based authorization, and UC for the reputation system), the communication between the reputation system (section 3.3) 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 within the ecosystem. The authorization component will store the policies received and, whenever a new reputation score from an entity is received, it will translate that policy to a network authorization rule.
Depicting a common flow, when a device tries to communicate to an internet service, the request will be routed through an operator network (TRU's, in this case), and the authorization will be applied there, in the network core, before the data arrives the service on the internet. If the device policies define that it is authorized to communicate with that service, the request will then be sent to the intended destination.

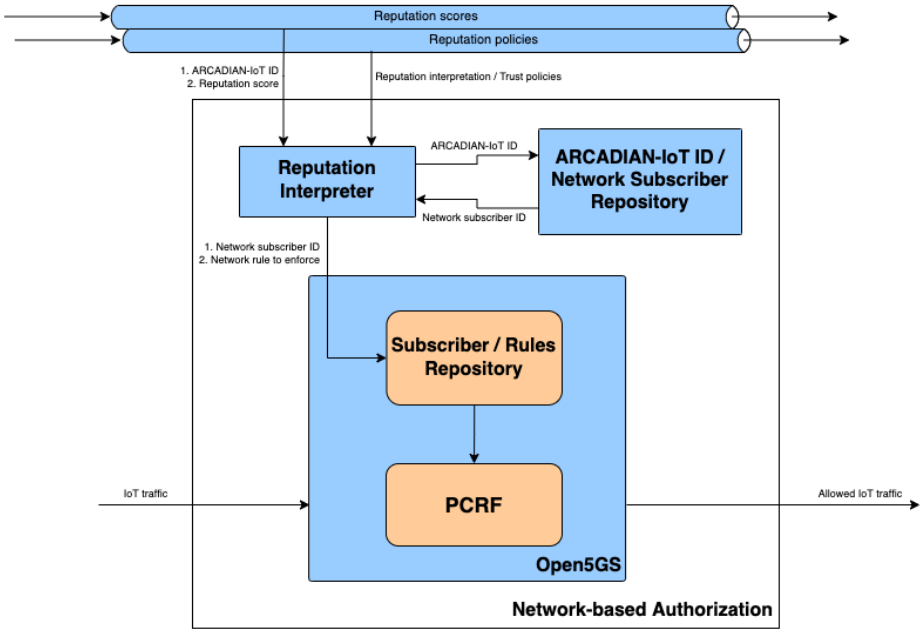

Figure 15 - Reputation-Authorization integration

In Figure 15, it can be observed the two topics related to the integration of the reputation-related events in the Authorization component, one for the reputation policies and another for the

reputation score of a given device. The publishing of these events is responsibility of the reputation system. When a new score is received by the *Reputation Interpreter*, a subcomponent of the network-based authorization, it will access an ID repository in order to translate the ARCADIAN-IoT identity into a network identifier (e.g., an IMSI). Finally, and using the API provided by Open5GS, the Interpreter submits a network rule (based on the policies received) to the network identifier that was just translated. Open5GS, which will be further explored below, is the network testbed being used for research purposes in ARCADIAN-IoT. In Figure 15 is just represented the PCRF (Policy and Charging Rules Function), the key software component that will, in real-time, determine the policy rules to apply in the network. In this case it will apply the novel trust-based rules generated automatically according to the entities trustworthiness.

The second feature of the network-based authorization leverages TRU's eSIM ecosystem and is depicted in Figure 14. Based on devices trust level changes, the authorization component requests an OTA service to securely inform the device secure element of those changes. The communication is secured according to GSMA Security Accreditation Scheme (GSMA-SAS)[11]. For the moment, we're assuming that the *Reputation Interpreter* is the subcomponent that triggers this request, because it has the information needed for such. Reputation changes from a device that was found to be trustworthy to not trustworthy, as well as the opposite, trigger actions of ARCADIAN-IoT eSIM security applet for Device Self-Protection or Device Self-Recovery (details purposely omitted for IPR protection analysis).

Adding trust-related policies to a core network and Open5gs

In today's network architectures, authorization, policy, and billing are already a focus point. 3GPP's Policy and Charging Control (PCC) architecture[12] provides access, resource, and quality of service (QoS) control[13] to mobile networks. Two components of this architecture are the Policy and Charging Rules Function (PCRF) and the Policy and Charging Enforcement Function (PCEF).

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 that require a dynamic PCC[14].

3GPP's PCC architecture describes an AF as "an element offering applications that require dynamic policy and/or charging control over the IP CAN (IP Connectivity Access Network) user plane behaviour." The Application Function extracts session information and media-related information from the application signalling and provides application session-related information to the PCRF using the Rx[15] 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 messages[15].

[11] https://www.gsma.com/security/security-accreditation-scheme/
[12] https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=810
[13] https://www.netmanias.com/en/post/techdocs/10997/lte-pcrf/policy-and-charging-rules-function-pcrf-in-lte-epc-core-network-technology
[14] https://www.juniper.net/documentation/us/en/software/junos/subscriber-mgmt-sessions/topics/topic-map/3gpp-policy-charging-control-provisioning-accounting.html
[15] https://www.netmanias.com/en/post/techdocs/10997/lte-pcrf/policy-and-charging-rules-function-pcrf-in-lte-epc-core-network-technology

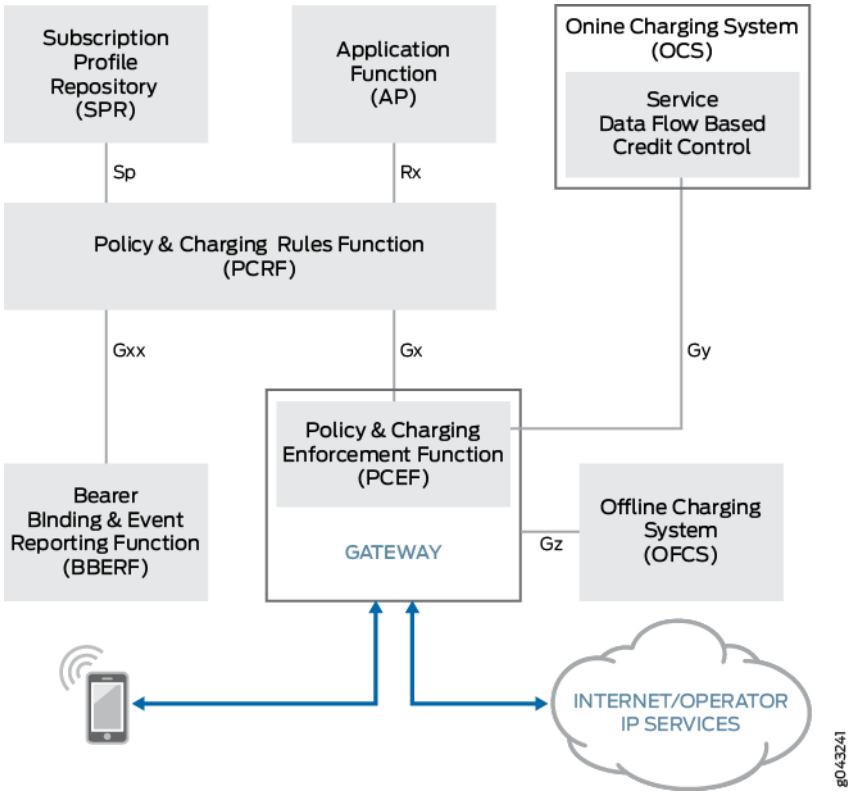

Figure 16 - 3GPP's PCC Architecture overview[14]

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 PCRF[16].

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 PCEF[17] and can be activated, modified, and deactivated in runtime.

In the context of ARCADIAN-IoT, PCRF/PCEF solutions can be leveraged to efficient and dynamically route and prioritize network traffic[13] as a means of providing trust-based authorization inside the network.

PCRF/PCEF uses policy-based authorization, however, as seen in[18], it is possible to build a mixed authorization system that mix static policy-based authorization with dynamic reputation-based authorization. This system combines the advantages of both systems 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 on. The main aspects considered were the presence of an API that allowed us to manipulate the subscriber information and their policy, but also, if possible, a free and open source solution.

[16] https://www.3gpp.org/more/1629-rx_interface
[17] https://www.netmanias.com/en/?m=view&id=techdocs&no=11863
[18] http://rewerse.net/publications/download/REWERSE-RP-2005-116.pdf

Nowadays there are a couple of solutions that fit this criterion, including Open5GS, Magma, srsEPC, to name a few. The current choice, as hypothesis for the first prototype 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.

3.2.2.1 Technical analysis

The vision over the features of the network-based authorization component seems 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 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 in the envisioned IoT scenarios. The research focus is, therefore, for the moment, in the novel component that allows the integration between the reputation system, the PCRF/PCEF present in the network testbed, and the network OTA services certified by GSMA-SAS.

However, the use of Open5gs as testbed still raises some doubts, particularly for using in demonstration phases, with real devices located in real networks. TRU's R&D team is researching a solution for this challenge with the company most senior networks professionals.

To start validating the hypotheses formulated, a first prototype (Authorization Prototype 1) has been developed with the following characteristics:

1. Network testbed: Open5gs
2. One to two virtual devices in the network
3. 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.

Table 6 depicts the status of the current network-based authorization component, including the network testbed.

#	Subcomponent	Brief description[19]	Prototyping status
1	**Network testbed**	Allows to integrate the network-based authorization component for research purposes, including for the demonstration scenarios integrating real devices in real networks. Includes the needed network core elements, namely a PCRF and a PCEF.	First prototype (Authorization prototype 1) done and unit testing performed in March-22. Unit testing showed issues in applying new policies in real time, without a hard reset –

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

2	**Network-based authorization – reputation interpreter**	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 (IMSI); Automatically generates PCRF/PCEF rules according to the trust-based policies and each entity reputation score; Upon a policy related to a reputation change, triggers an OTA to ARCADIAN-IoT eSIM security applet to perform device self-protection or self-recovery (depending on the reputation change).	hypothesis of correction measures formulated and ongoing. (With the hard reset) The trust-based policies described before for the first prototype (Authorization prototype 1) worked as expected. A demo of the current prototype exists and can be provided.

Table 6 – Current status of the network-based authorization component

As can be seen in the table above, the major challenge found in the development of the first prototype seems to be related with the network testbed environment. PCRF/PCEF solutions seem to allow the enforcement of rules according to the needs of ARCADIAN-IoT network-based authorization component. This assumption will prove to be false if the research shows not to be possible the enforcement of the rules in real time or near real time in PCRF/PCEF, factor that is critical to ensure the security and privacy of the data and of the devices. At the moment, TRU's R&D team is working with the Open5gs community to understand the issue and candidate solutions exist and are being implemented and tested.

3.2.3 Current resources

Open5GS provides various resources, which have been source of knowledge for the current work:

- Website[20] with blog, tutorials, and other documentation.
- GitHub repository[21] with source code, issue tracker and discussion board.
- Discord server with a community chat room (invitation link is provided via GitHub's readme).

3.2.4 Future work

As future work we envision the following, which is associated with an enhanced prototype for the component:

- Finish the research on the challenge related with real time application of novel trust-based policies.
- Deepen the research in enforcing authorization policies in the flow that comes from internet services to the devices (at the moment most of the focus was in controlling devices traffic flow to the internet).
- Research on the triggering of the OTA services to devices secure element (ARCADIAN-IoT eSIM security applet): when to do it and what to send (the how, is expected to be through certified GSMA-SAS OTA services, to ensure secure communication).
- Incorporate a more granular set of trust-based policies.

[20] https://open5gs.org/
[21] https://github.com/open5gs/open5gs

- Research in the integration of the self-recovery routes in the network-based authorization component.
- Testing with a real IoT device, in a real network, connected to the internet through the network testbed, and with trust-based policies being enforced.

3.3 Reputation System

3.3.1 Overview

3.3.1.1 Description

The Reputation System component in ARCADIAN-IoT determines the reputation values – score associated with the entities in the ARCADIAN-IoT framework – persons, devices and services. The reputation score represents the trust information regarding a certain entity, and such information is built based on data received from other entities regarding interactions. 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 [1]:

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 act as input for the reputation system. Such interactions can be intra- 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 point of failures.

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:

- Number of messages analysed per unit of time: Messages indicating interactions between entities
- Time required to determine reputation
- Types of entities supported by the reputation system, at least 3 types

3.3.2 Technology research

3.3.2.1 Technical analysis

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 users[22].

Of particular interest in ARCADIAN-IoT is the consideration of the beta distribution[23,24], that can consider two types of events:

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

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

In order 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.

$$r = v.w \; s = w(1 - v)$$

The values of r and s are employed to determine α ALPHA and β 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)[25] 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 chosing the services that

[22] 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

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

[24] 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

[25] 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.

will be ranked, in case the dominance information is not objective.

3. Considering the DAG determine the rank of services, considering the following equation:

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

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

3.3.3 Current resources

Besides the references presented in the previous subsection, the following specifications are or interest:

- ETSI GS NFV-SEC 0002 – Security in Network Function Virtualization
- IETF Repute working group with the recommendations of RFC7070, RFC7071 and RFC7072

The reputation system also uses the following resources:

- Private repository for development and integration activities
- Instantiation of reputation system in a private server for development purposes

3.3.4 Future work

The future work of the reputation system is the following:

- Gather inputs from other components, considering the interfaces with the reputation system documented in D2.5
- Define, and agree with remaining partners the message bus that will be used and a pre-list of topics to be used by components exchanging information with reputation system.
- From the input of domain A validate the approach to determine reputation, considering the DRBR approach.
- Test and validate the performance and efficiency of the diverse approaches to determine reputation.

3.4 Remote Attestation

3.4.1 Overview

3.4.1.1 Description

The remote and functional attestation 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 hardware-based Attestation, with the ability to leverage Root-of-Trust using a Trusted Platform Model – (e.g., eSIM, or crypto chip).The Remote Attestation component will consider the heterogeneity regarding devices' capabilities, being designed with efficient remote integrity verification and challenge-response mechanisms, while being aligned with the IETF Remote Attestation Procedures (RATS) working group – 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 RATS-based remote attestation for IoT services is also scoped, towards enhancing service authorization functionality (i.e. ensuring service trustworthiness) but it's support is still

subject to future research.

3.4.1.2 Requirements

The following requirements have been specified in D2.4 [1]:

Requirement 5.4.1 - **Attestation pre-installation:** The (IoT) device must have or enable 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 protected environment such as TPM to receive 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** - Confidentially of Evidence and Attestation Results should be protected via encryption.

3.4.1.3 Objectives and KPIs

The work to be pursued for the attestation system can be decomposed into some key objectives:

- Supporting Remote and Functional Attestation providing Root of Trust mechanisms in TPMs
- Supporting Attestation involving multiple verifiers

While the associated KPIs include:

4.1 - Remote attestation (i.e., Attester component) supporting at least one of ARCADIAN-IoT TPMs;

4.2 - Remote attestation (i.e., Attester component) supporting at least 2 types of ARCADIAN-IoT devices/platforms.

Additionally, the component is expected to contribute to the following overall ARCADIAN-IoT objective and associated KPI:

- **Objective 4**: Provide distributed and autonomous models for trust, security and privacy – enablers of a Chain of Trust (CoT)
    - **KPI 4**. Availability of trust evaluation models for heterogeneous entities (devices, services, persons)

3.4.2 Technology research

3.4.2.1 Technical analysis

Background on Remote Attestation

The generic Remote Attestation process shown in Figure 17 is done according to the following logic steps:

1. The untrusted device (as attester) produces claims. Claims contain encrypted information about the device.
2. Claims are then digitally signed to generate Evidence by the Attester.
3. Evidence is sent to the verifier that will appraise it via appraisal policies.
4. The Verifier appraises Evidence via appraisal policies and creates the Attestation Results

to support Relying Parties in their decision process.

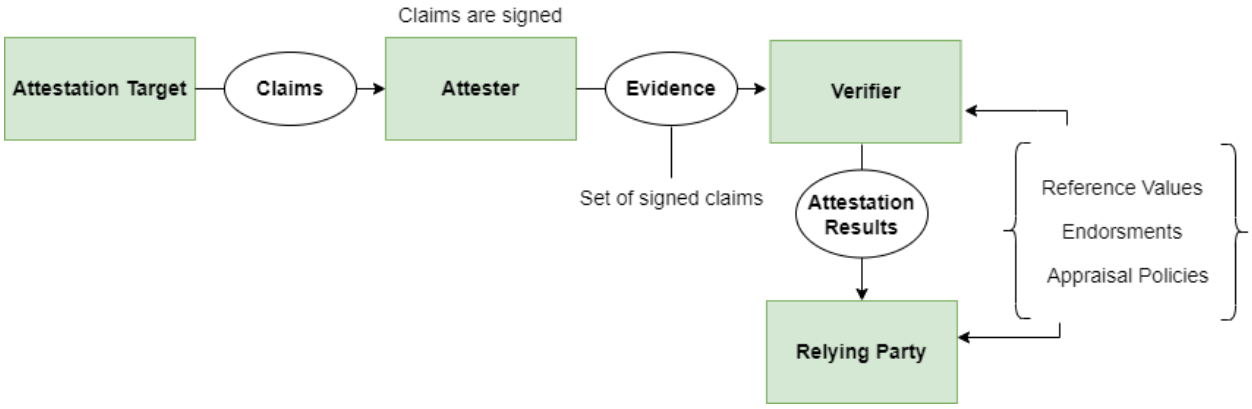

Figure 17 Generic Attestation Procedure

Moreover, three main interaction models are considered regarding Remote Attestation: 1) challenge/response, where the Verifier triggers the remote attestation process; 2) uni-directional, which can be initiated by either the Attester of Verifier, or streaming, in which there's subscription state is kept between the Verifier and the Attester.

Within the challenge/response interaction model, there are 2 reference models, differing on who initially receives the Evidence. In the **Passport Model** (Figure 18, left side)**,** an Attester conveys Evidence to the Verifier, which compares the Evidence against its appraisal policies. The Verifier then gives back an Attestation Result which the Attester treats as opaque data, I.e., the Attester does not consume the Attestation Results but might cache it, before sending them to the Relying Party, which compares this information against its appraisal policies.

In the **Background check model** (Figure 18, right side)**,** an Attester conveys Evidence to a Relying Party, which simply forwards it on a Verifier. The Verifier then compares the Evidence against its appraisal policies and returns an Attestation Result to the Relying Party which compares them to its appraisal policies.

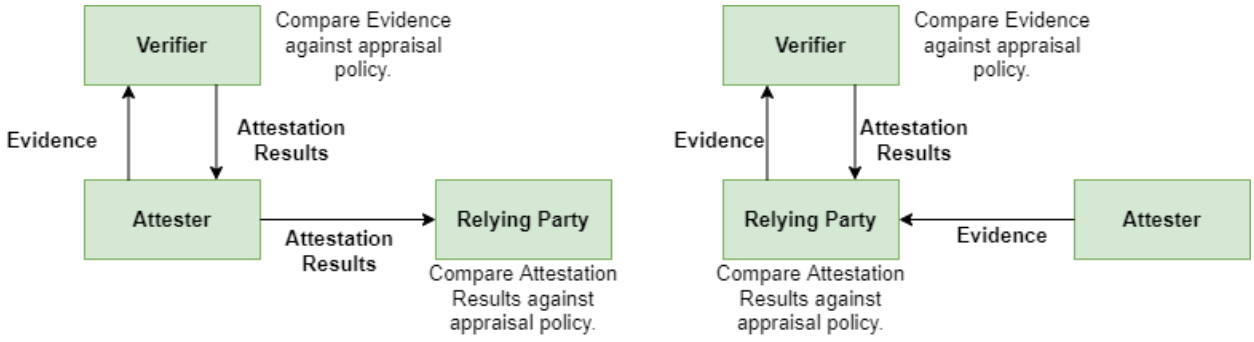

Figure 18 Passport Model vs Background-check model

One potential variation of the background-check model occurs when the Relying Party and the Verifier are collocated, I.e., perform functions on the same machine. In this case, there is no need for a protocol between the two.

It is important to define in which situations Attestation is triggered and by who. The goal is to trigger attestation on enough occasions that ensure that the system can be secure but avoid triggering it so many times that it might introduce excessive overhead. Some potential triggers include a device boot or system reset, a request for accessing a specific data or service, or watchdog timer reset (as per Requirement 5.4.3 above). Other use cases where remote

attestation can be applied include [1]a device desiring access to a network, an entity desiring to retrieve confidential data, a device or application wishing to control physical equipment, a device needing an update, among others. Another interesting possibility, proposed in PIV[2] is performing attestation when a device tries to join the network or when an intrusion detection mechanism flags the device as potentially compromised.

Claims are pieces of asserted information about the remote device and make up the usual structure of Evidence. Some possible information that can be considered a claim, includes the proof of the make and model of the device hardware (HW), proof of the make and model of the device processor, particularly for security-oriented chips, measurement of the software (SW) running on the device, configuration, and state of the device and the Environmental characteristics of the device such as its GPS location.

Common formats for passing security information include JWTs [RFC7519], CWTs [RFC8392], and X.509 certificates[26].

The Evidence and Attestation Results reveal a lot of information about the internal state of a device as well as potentially any users of the device. For example, knowing that a device is running a weak version of firmware provides a way to aim attacks better.

In some cases, an attacker may be able to tamper with the Evidence. For example, an attacker might be able to infer the value of specific Claims if it knew that only certain values were accepted by the Relying Party.

For those reasons, it's fundamental to maintain a secure attestation environment. These are some **security considerations** that should be considered[27]:

- Maintaining an isolated and protected Attesting environment.
- Confidentiality protection of the Attesting Environment's signing key.
- The keys generated in the factory, whether generated in the device or off the device by the factory should be generated by a Cryptographically Strong Sequence.
- Attestation key provisioning must ensure that only valid attestation key material is established in Attesters.
- End-to-End Encryption, Denial of Service protection, Auditing, Access Controls, logging…

Attestation procedures are vulnerable to several types of attacks identified in the literature[28], such as **Denial of Service Attack (DoS) or Time-of-check-to-time-of-use Attack (TOCTTOU)[29].** A DoS attack is characterized by an attempt by attackers to prevent the normal use of a service by flooding the network with requests to that service. This type of attack can be targeted to the Verifier and can be prevented by monitoring the network traffic and having blacklists and can be mitigated by ensuring server redundancy and having a DoS Attack plan. A TOCTTOU Attack is a File-Based Attack in which the Attacker explores system errors, and it might lead to privilege escalation, unauthorized access to resources, such as read and write access as well as avoiding log and audit controls. It's important to protect the system against this type of attack to prevent the information tampered with. TOCTTOU can be prevented by adopting variants of common file system calls that operate on file handlers rather than file names and by using libraries for tracking file descriptors and ensuring correctness.

[26] https://datatracker.ietf.org/doc/draft-ietf-rats-architecture
[27] https://datatracker.ietf.org/doc/draft-ietf-rats-reference-interaction-models/
[28] Boyu Kuang, Anmin Fu, Willy Susilo, Shui Yu, Yansong Gao, "A survey of remote attestation in Internet of Things: Attacks, countermeasures, and prospects", Computers & Security, Volume 112, 2022, 102498, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2021.102498.
[29] https://cwe.mitre.org/data/definitions/367.html

Remote Attestation support in ARCADIAN-IoT

In order to implement Attestation procedures in ARCADIAN-IoT, the Attester entity, located in the IoT device, will be supported by TPMs - either eSIM or encryption device - for providing root-of-trust, and by the Hardened Encryption library to ensure confidentiality of the attestation evidence.

Claims (e.g. hardware model, timestamp, token ID) will be collected and then, as part of Attestation Environment functionality, encrypted (via the Hardened Encryption library), before the associated payload is signed and generated a signed hash, i.e. attestation evidence. The Hardened Encryption – Key Management subcomponent - is expected to provide to the target Verifier(s) the secret keys (as endorsements) required to (functionally) decrypt the necessary attestation claims, with different verifiers selectively decrypting only the evidence they are supposed to appraise via their policies. Another form of Verifier's endorsement would be the public key and associated data (e.g. public key algorithms and associated parameters, negotiated prior to the process), provided by the eSIM for RoT signature validation, which ensures the integrity of the sender of the information (proof of the true sender of the data). Upon the key pair generation by the eSIM, it securely sends the public key to the Verifier, which stores it (as Attester's key material) in its trust anchor store – required so that the Verifier can trust the attested devices.

The Attestation results will support the Chain of Trust by feeding the Reputation System with crucial information for assessing IoT devices trustworthiness. After receiving the attestation results, the Reputation System processes them, and may review the device's reputation according to its policies, either increasing (e.g. in case of remote attestation after booting the device for the first time after recovery) or decreasing (in case of attestation failure) the device's reputation score. Remote Attestation takes an important role in ensuring device trustworthiness. While it is subject to be aligned with the Reputation System and enforcement components (e.g. network authorization or recovery), some current possibilities regarding trust policies include the following:

- if the device reputation is between zero and a certain threshold (e.g. 0.5), a device reset/recovery is triggered.
- if the reputation is set to zero, the Authorization component should blacklist the device, disabling it from accessing the network.

The high-level representation of the remote attestation process in ARCADIAN-IoT is depicted in Figure 19.

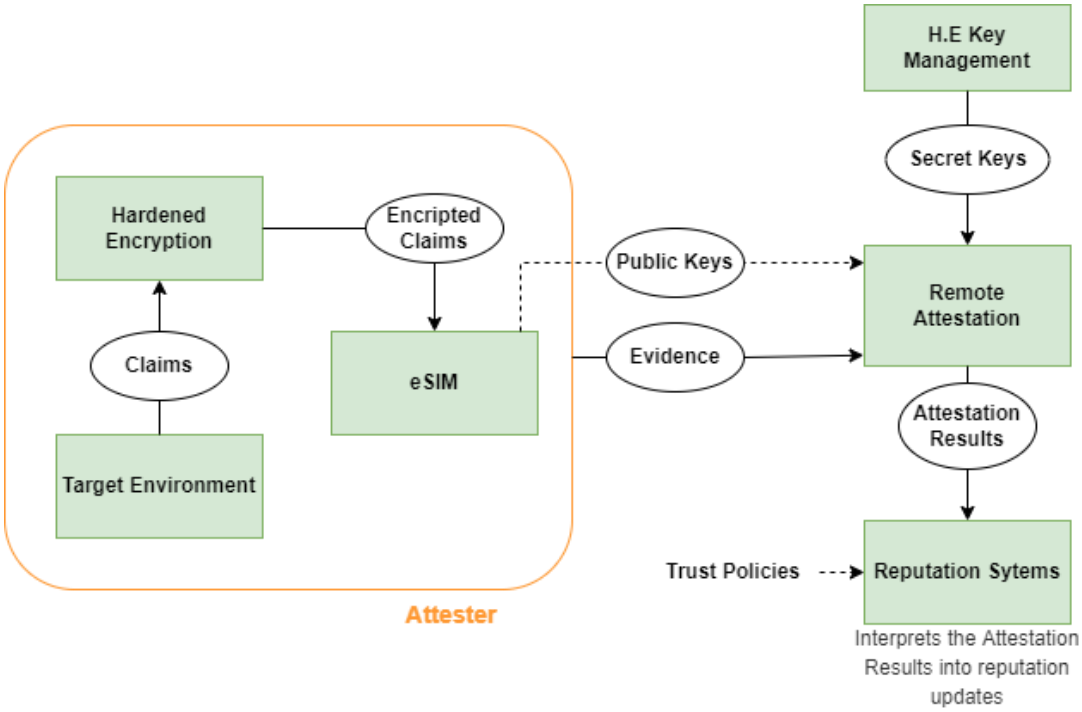

Figure 19 High-level diagram for Attestation in ARCADIAN-IoT

3.4.3 Current resources

There are no resources currently available to public at this moment. We aim to explore existing relevant resources such as Fraunhofer's challenge response implementation for TPMs[30], and, considering we're still finalizing the design phase. We intend to publicly release the source code of the remote attestation component once sufficient progress is reached.

3.4.4 Future work

We're currently at a stage where several design decisions need to be made. Some of the research decisions (to be addressed by June 2022) include:

- Stabilize the exact list of potential triggers for attestation procedures addressed in the project; some possibilities have been discussed, additional ones may still be explored such as having Behaviour Monitoring trigger the Remote Attestation;
- Decide the "format" of Evidence, sent from eSIM as an attesting environment to the Verifier. The Entity Attestation Token draft[31] points out some options which will be considered, namely CBOR Web Token (CWT), JSON Web Token (JWT), Unprotected CWT Claims (UCCS) or Unprotected JWT;
- Whether to collocate or not the Verifier and Relying party entities; opting for such collocation would discard the need for a protocol to carry the Attestation Results; this is at the moment the probable option, as it would allow to give higher focus to the support of multiple verifiers; nevertheless, such decision is tied to decision on the final appraisal policies and evidence to consider;
- The nature and semantics of the Attestation Results: the options include delivering a binary (attestation success / failure) type of result, which implies higher importance of the

[30] https://github.com/Fraunhofer-SIT/charra
[31] https://datatracker.ietf.org/doc/draft-ietf-rats-eat/

Verifier's evidence appraisal policy, or more sophisticated information such as the Evidence Set plus some reporting;

- Which should be the appraisal policies for evidence and for the attestation results (related to the previous point);
- Define exact scope of service attestation, aiming to fulfil ARCADIAN-IoT full scope and objectives.

After reaching a consensus about the design and architecture of the component, until D4.2, we intend to initiate implementation work, focusing the Attester functionality (I.e. the capability of generating claims, claims formatting, and evidence transmission). For this, we will consider existing resources such as Fraunhofer's challenge response implementation. In parallel, during this period, work sessions with both TRU and XLAB will be critical, in order to discuss and assess approaches for integrating with eSIM applet and Hardened Encryption libraries at the Attester, respectively. We're aiming at completing this stage by M24, with preliminary results being reported in D4.2 (M20).

Finally, and until D4.3 delivery, we intend to focus on the verification functionality (i.e. capability to appraise evidence and generation of attestation results, and relevant appraisal policies with respect to the target use cases), where alignment with the Reputation System will be critical. This stage will also address the domain and hardware-specific requirements (e.g. drone vs smartphone) and associated implementation and testing.

Figure 20 Timeline for Attestation System implementation

4 RECOVERY MANAGEMENT PLANE

The research activity related to the Recovery Management plane is part of of Task 4.3, which addresses two components: **Self-Recovery** and **Credentials Recovery**.

4.1 Self-recovery

4.1.1 Overview

4.1.1.1 Description

The Self-recovery component is composed of a storage server, that exposes a REST API via HTTP/S and client-side (on-device) scripts, that allow devices interface with the storage server and store and retrieve backups. The types of data that will be stored will vary from device to device, ranging from configurations that are required for the device to operate normally, system logs 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 predicated on its reputation score, meaning a device with compromised security will need to go through the process of 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

The primary objective of the Self-recovery service is to provide an automated recovery mechanism of data and credentials for devices that were affected by attacks, bugs, theft/loss or software updates that introduce incompatibilities. Additionally, advanced techniques like attribute-based encryption (results of Hardened Encryption task) will be used to enable policies defining who can decrypt the data and what can be decrypted.

The evaluation KPI defined for the Self-recovery service is:

- The recovery process is successful if the application/process/device is running as expected.
- 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.

4.1.2 Technology research

eSIM and network-based authorization role in device's self-recovery

eSIM, particularly the novel security applet stored in the secure element, and the network-based authorization component participation in device's self-recovery were target of research in the current reporting period, resulting in a vision that will be prototyped in the forthcoming one.

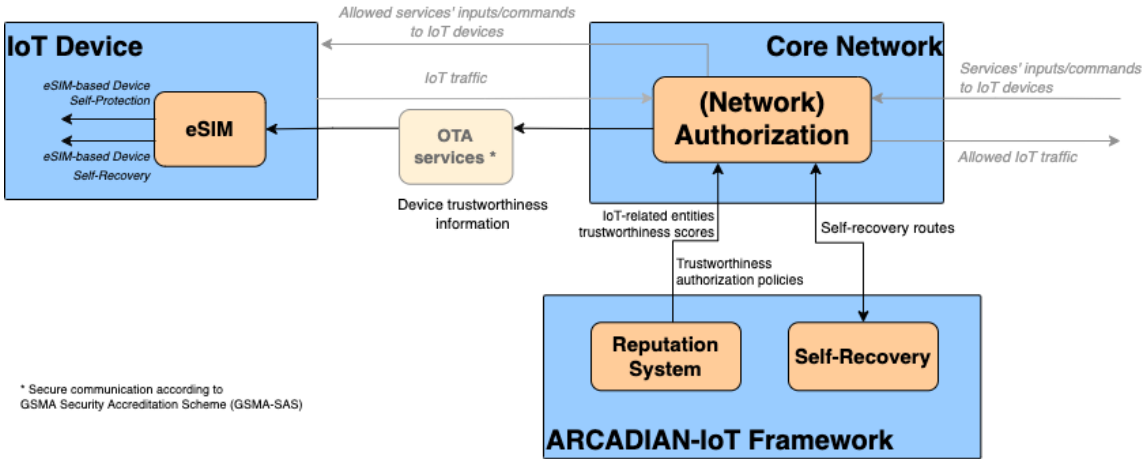

Figure 21 - eSIM and network-based authorization in device's self-recovery

Figure 21 depicts the specificity of the hypotheses formulated, which can be subdivided in two contributions. The first one consists of leveraging the network-based authorization component ability to enforce rules/policies in the network core (between IoT devices and the Internet) to automatically enforce rules of self-recovery to the targeted devices. This means that devices that are selected/targeted for recovery, given that likely are compromised in terms of security or privacy, will only be able to access self-recovery services, nor being allowed other communications until their successful recovery. This avoids the compromised device to continue its unsecure actions (or for being accessed externally), while allowing it to perform the programmatic self-recovery actions that are network dependable.

The second contribution joins the network-based authorization with the eSIM security applet stored in the novel ARCADIAN-IoT eSIM profile (further details in D3.1, Hardened Encryption section). The Network Authorization component is planned to receive, in real time or near real time, updates on the devices trustworthiness level and the related authorization policies from the reputation system. When a device whose trustworthiness was in a level that determined that it was compromised (policies and levels to be defined within the reputation system research), and its reputation has recovered to a safe level (e.g. because self-recovery actions successfully took

place) the Network Authorization component distributes the updated trustworthiness information to the device secure element. ARCADIAN-IoT eSIM profile should be ready to receive this information and perform specific actions of automatic operational recovery[32]. The communication happens securely, using over-the-air services accredited by GSMA-SAS[33].

Storage backend

Based on use-case feedback regarding the requirements related to size and types of data, an appropriate storage backend will be selected. A proof-of-concept version of the storage server currently uses Openstack Swift[34] for file storage. Other considerations are XtreemFS[35], CEF[36], IPFS[37], or simple blob storage of traditional relational databases. Each will be evaluated for ease-of-integration, applicability to size and data type requirements and performance.

REST API technology

The proof-of-concept version of the storage server is implemented as a NodeJS service, while the API is documented with Swagger, OpenAPI[38] v2. Proper implementation of the server will move to a more robust language, the front-runner is Golang, as it will ease the efforts of integration with the results of Hardened Encryption.

4.1.2.1 Technical analysis

Storage backend

The choice of the storage backend will be determined alongside with an analysis of the requirements of integration with the authentication and reputation systems. If those requirements necessitate the use of a relational database by the storage server, it will also be used as a blob store for backups, otherwise, one of the file-specific services above will be evaluated and the most appropriate will be selected.

REST API technology

Golang is the most appropriate language to use for the implementation of the storage server due to the fact that Hardened Encryption is working with Golang already. A REST framework that supports OpenAPI v3 will be selected, since it provides self-documentation of the interface.

4.1.3 Current resources

The proof-of-concept versions of the self-recovery server, client scripts and a demo will be publicly available soon after the publication of this deliverable.

4.1.4 Future work

Moving from a proof-of-concept to a prototype version of the Self Recovery component will include several steps of technical implementation, refinement and integration. An overview of the use-

[32] Specific processes removed for allowing future IPR protection measures
[33] https://www.gsma.com/security/security-accreditation-scheme/
[34] https://wiki.openstack.org/wiki/Swift
[35] http://www.xtreemfs.org/
[36] https://ceph.io/en/discover/technology/
[37] https://ipfs.io/
[38] https://swagger.io/specification/

case requirements regarding data types and volumes that require backing up and storage, will guide the selection of the storage backend. Further refinement of the role of eSIM in the communication flows between the Self Recovery component and the platform will be happening in parallel with integration with the ARCADIAN-IoT authentication and authorization services.

Use-case requirements will also dictate the development of the encryption proxy and pluggable storage, depending on data privacy concerns and the capabilities of the IoT devices used.

Another point of investigation will also be the interaction with the Self-protection and Reputation components. Device Self-protection will play a major role in determining the need for recovery, while the success or failure of the recovery process will influence the device reputation score.

4.2 Credentials recovery

4.2.1 Overview

4.2.1.1 Description

The recovery of credentials is the first and necessary step to trigger a data recovery mechanism. The secure recovery of credentials is vital 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 and to the functional encryption keys with self-sovereign Identity, which relies upon Decentralized identifiers and Verifiable Credentials components as described in the Trust Pane of section 3.

To avoid manual recovery of the credentials, various techniques will be evaluated and the most suitable one will be implemented. One technique might be to allow the device to re-establish the credential through the intervention of a quorum of trusted entities. These trusted entities are user-defined and can be individuals or institutions, like banks and credit unions.

4.2.1.2 Requirements

A recall of the requirements defined in ARCADIAN-IoT D2.4 [1] with further supplemental information is detailed here.

- Requirement 7.2.1 – Credentials recovery mechanism
  - To recover lost, compromised or corrupted credentials for an SSI Agent or Wallet. Analyse also the recovery of network credentials from network operator for authenticating devices/persons in third parties.

4.2.1.3 Objectives and KPIs

The primary objective is to provide the secure recovery of credentials as the first and necessary step to establish the trust before the data recovery mechanism is triggered.

Additionally, it will be researched on how to securely restore the private data (authentication / authorization credentials) that was stored in the hardware of the device (eUICC) before the need of recovery.

KPIs defined for Credentials Recovery are listed below:

- Support VC Recovery operations after security/privacy incidents
- Investigate two means of credential recovery and implement at least one

4.2.2 Technology research

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 and encryption keys. First, we specify the authenticated access based on Self-Sovereign Identity considering also potential limitations of constrained devices and then analyse possible solutions to support credential recovery, to be used by persons and devices to access the Self-Recovery component.

4.2.2.1 Self-Sovereign identity Authentication and support for Authorisation

Access to backup files will be protected by Self-Sovereign Identity so that only those with the corresponding pre-registered DID and/or Verifiable Credential can be authorised to access the backup.

It is further specified that a critical component is the functional encryption key server to also be protected by SSI means so that only authorised users can get access to the functional keys used to decrypt backups.

Therefore, the backup server and the functional encryption key server that are part of the overall Self-recovery solution outlined in section 4.1 is able to be protected by SSI for authentication and support of authorised access.

Note:
It is expected that the SSI Wallets will use self-generated privacy preserving peer DID method[39] in a pairwise fashion (not anchored on-chain), whereas IoT Devices will use public DIDs that will have their trust anchored on a permissioned blockchain.

To support authenticated access by users such as administrators and operation team members to Self-recovery, this will make use of the Decentralised Identifiers and Verifiable Credentials components described in sections 2.1 and 3.1 respectively, by presenting a previously issued Verifiable Credential. The frontend of the Self-recovery component should then generate a self-describing access token based on the account claims presented in the VC and this will be used in further http requests as a bearer token for authorising the access.

This is depicted in the following figure.

39 https://identity.foundation/peer-did-method-spec/

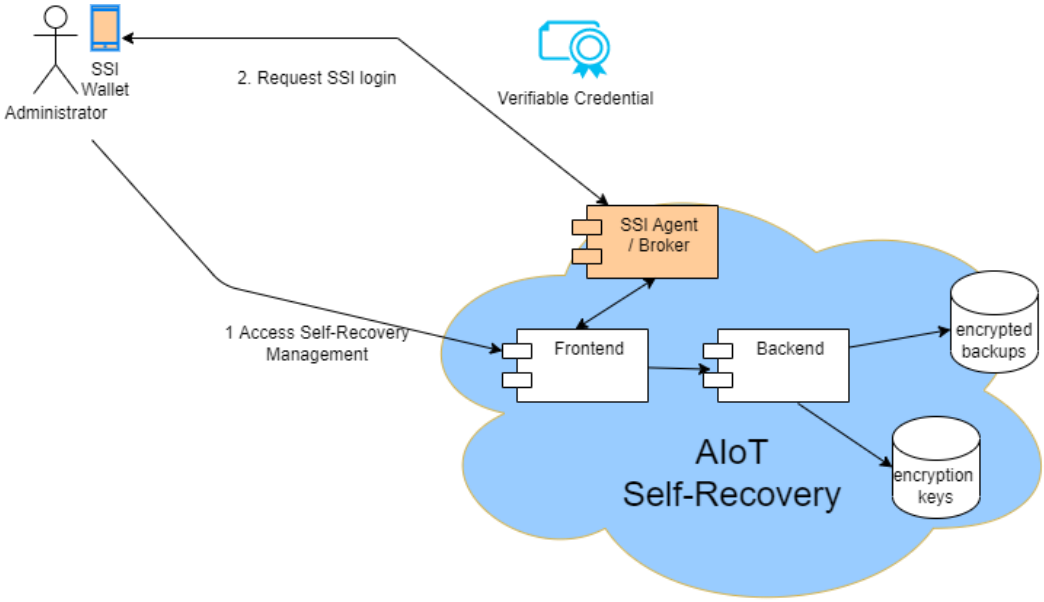

Figure 22 ARCADIAN-IoT Verifiable Credential access to Self-Recovery

In the case of constrained IoT devices, including those that are not able to support a full SSI Agent stack, and hence not able to handle Verifiable Credentials, it is proposed to make use of an ACADIAN-IoT Device Trust Registry (ARCADIAN-IoT-DTR) that registers the public DIDs of these IoT Devices. Note that DIDs in ARCADIAN-IoT are only able to be issued by authorised ARCADIAN-IoT organisations as described in section 2.1 and it will be the same organisations that only are able to write to the ACADIAN-IoT Device Trust Registry. To support this, it is therefore also needed to implement an ARADIAN-IoT Organisation Trust Registry (ARCADIAN-IoT-OTR) with the list of trusted organisations in the ARCADIAN-IOT deployment. The figure below depicts this high-level architecture approach, noting that the devices would have to be first registered with a Self-recovery account by an administrator registering the DIDs for each device. During the registration it would be specified that the device does not support VCs and would need to check the ARCADIAN-IoT-DTR registry.

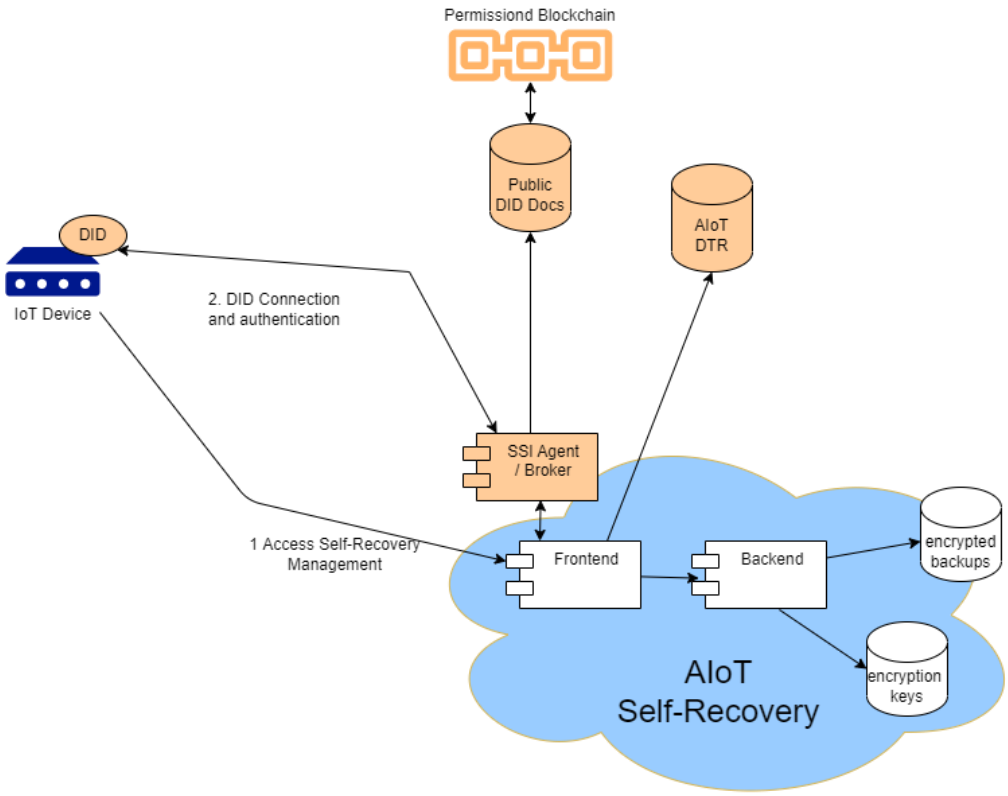

Figure 23 ARCADIAN-IoT DID Access to Self-Recovery

It is important to note that the device needs to support DIDs with the DIDCOMM protocol as specified in section 3.1.

As no VCs are supported for IoT Device accounts, DID authentication is carried out with a challenge response requested by the Self-Recovery frontend upon the DID connection being established.

4.2.2.2 Credentials Recovery

Credential recovery considers the scenario where a user´s or IoT device´s data (including credential SSI Wallet or from an IoT device) was somehow corrupted or wiped and the user or device is attempting a recovery of credentials and 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.

Support for automatic recovery of credentials for mobile and IoT devices are described below.

Self-Recovery Account 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 are aimed to be supported:

- 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 Self-recovery component.

Note:
If the peer did was compromised or lost a new peer did or updated version would be created for the wallet.

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

To support an automated process for restoration of a device it is needed for the device to connect to the Self-recovery component with their public DID and perform DID authentication where the DID is also checked to be in the trusted ARCADIAN-IoT DTR registry. Using a public DID anchored on the ARCADIAN-IoT Permissioned Blockchain ensures that the device is the holder of the private key associated to the DID.

In the case of IoT devices that lost their DID or was compromised the DID DOC would have to be first updated / recovered as described in section 2.1 with the IoT device updating its associated private key.

4.2.3 Current resources

This is a new development and as such there is no existing resources supporting Credentials Recovery in itself.

4.2.4 Future work

The next step is to elaborate the design and agree the interwork with the Self-Recovery component and WP5 Pilots before starting its implementation.

5 CONCLUSIONS

The main thrust of this technical report presents the preliminary findings of the research activities and State-Of-The-Art for the ARCADAIN-IoT Vertical Plane components. The report further describes each vertical plane component itself and the current status of the research while also considering partner background, before outlining future work in the development of the components to meet the requirements and use cases.

As part of this work, the components identify the use of specific components in the horizontal plane of ARCADIAN-IoT framework (for example Decentralized Identifiers use of the blockchain). Therefore, next steps will also plan to more closely collaborate with WP3 to ensure that the vertical plane components requirements on horizontal components are fully considered.

Additionally, once the research is concluded and the design is formally agreed, the components being developed will specify the interfaces not only to the Horizontal Plane components, but also into each of the domains where it will be integrated and deployed, in collaboration with WP5.

The resulting technologies developed within WP4 and integrated in WP5 will be released in an initial prototype on M20 and will be documented in the D4.2 report. During the testing subsequent updates and improvements are envisaged to each of the components, so that a final prototype will be released on M30, along with the accompanying D4.3 report detailing the final design of each component.

6 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 node.js implementation”, https://github.com/transmute-industries/sidetree-core/blob/master/docs/implementation.md, 2022

[17] Transmute, “Sidetree Protocol Specification”, https://github.com/transmute-industries/sidetree-core/blob/master/docs/protocol.md

[18] https://w3c-ccg.github.io/did-method-key/, 2022

[19] https://identity.foundation/peer-did-method-spec/index.html, 2022

[20] https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Early+Adopters+Programme, 2022

[21] https://joinup.ec.europa.eu/collection/ssi-eidas-bridge/about, 2022

[22] Evernym, https://www.evernym.com/blog/bbs-verifiable-credentials/, 2022

[23] DIF, https://identity.foundation/didcomm-messaging/spec/, 2022

[24] Transmute, https://github.com/transmute-industries/sidetree.js/blob/main/packages/did-method-element/README.md, 2022

[25] https://github.com/uport-project/veramo, 2022

[26] https://veres.one/, 2022

[27] https://www.hyperledger.org/use/hyperledger-indy, 2022

[28] https://github.com/hyperledger/aries, 2022

[29] https://jolocom.io/, 2022

[30] https://github.com/mattrglobal, 2022

[31] https://github.com/spruceid/ssi, 2022

[32] https://github.com/iotaledger/identity.rs/, 2022

[33] https://github.com/alastria/alastria-identity, 2022

[34] https://github.com/decentralized-identity/interoperability/blob/master/agenda2021.md, 2022

[35] https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/EBSI+Verifiable+Credentials+Playbook, 2022

[36] https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0.html, 2022

[37] https://openid.net/specs/openid-connect-self-issued-v2-1_0.html, 2022

[38] https://datatracker.ietf.org/doc/html/draft-looker-jwm-01, 2022

[39] https://w3c.github.io/vc-test-suite/implementations/, 2022

