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

ARCADIAN-IoT

Autonomous Trust, Security and Privacy Management Framework for IoT

D3.1: Horizontal Planes - first version

Revision: v.1.0

Work package	3
Task	Tasks 3.1, 3.2, 3.3, 3.4, and 3.5
Due date	30/4/2022
Submission date	30/4/2022
Deliverable lead	RISE
Version	1.0
Partner(s) / Author(s)	RISE: Alfonso Iacovazzi, Han Wang, Shahid Raza IPN: Paulo Silva, Sérgio Figueiredo, João Rainho, Vitalina Holubenko

	UWS: Jose M. Alcaraz Calero, Qi Wang, Antonio Matencio Escolar, Ignacio Sanchez Navarro, Pablo Benlloch Caballero, Pablo Salva Garcia, Enrique Chirivella Perez, Ruben Ricart-Sanchez TRU: João Casal, Carlos Morgado, José Rosa, Tomás Silva, Ivo Vilas Boas XLAB: Tilen Marc, Benjamin Benčina MARTEL: Giacomo Inches BOX2M: Alexandru Gliga ATOS: Ross Little, Miguel Montero

Abstract

This public technical report constitutes the deliverable D3.1 of ARCADIAN-IoT, a Horizon2020 project with **the grant agreement number 101020259**, under the topic **SU-DS02-2020**. D3.1 is the first of a series of three deliverables planned for reporting the findings, achievements, and outcomes resulted from WP3 research activity.

The material in this document presents the main outcome of all tasks in WP3 during the first seven months of the work package (from M6 to M12). WP3 is dedicated to the technological development of the components in the Horizontal Planes of ARCADIAN-IoT framework for each use case defined in Task 2.1 (Use cases specification and planning) and before being implemented in WP5. WP3 is organized in five main tasks (from Task 3.1 to 3.5), each of which focusing on the definition and development of one or more components in the Horizontal Planes.

Keywords: ARCADIAN-IoT, Privacy preservation, Intrusion detection, Intrusion prevention, Self-healing, Self-protection, Hardened Encryption, Permissioned Blockchain, Cyber Threat Intelligence.

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	28/02/2022	Preliminary Template	RISE
V0.2	15/03/2022	Overview subsections	RISE, UWS, MAR, IPN, TRU, XLAB, ATOS
V0.3	11/04/2022	Technology research subsections	RISE, UWS, MAR, IPN, TRU, XLAB, BOX2M, ATOS
V0.4	13/04/2022	Updates in some subsections	UWS, XLAB, BOX2M, ATOS
V0.5	28/04/2022	Updates after internal revision	RISE, UWS, MAR, IPN, TRU, XLAB, BOX2M, ATOS
V0.6	30/04/2022	Finalization	RISE

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: Technical report, technical diagram, software, evaluation, etc.

EXECUTIVE SUMMARY

The ARCADIAN-IoT project aims to provide a novel framework to manage and coordinate cyber security functionalities in IoT systems. The framework is organized in multiple planes combined together in an optimized way to support the end-to-end services: three Vertical Planes for the management of identity, trust, and recovery, and three Horizontal Planes for monitoring and managing privacy of data, security of entities, and providing Permissioned Blockchain and Hardened Encryption technologies.

Deliverable 3.1 (Horizontal Planes - first version) is a technical report presenting the preliminary outcomes of the research activities performed around the components in the ARCADIAN-IoT framework that belong to the horizontal plans (Privacy, Security, and Common planes) and are being developed in Work Package 3 (WP3) of the ARCADIAN-IoT project. Every component is introduced by providing a brief overview of its internal architecture and recalling respective objectives and target Key Performance Indicators (KPIs) before presenting a high-level description of the main initial outcomes related to the design and development of the component itself. The planned future work and current resources publicly available (if any, i.e., software, prototype, etc.) are then described for each component.

The work performed in WP3, and described in this document, is organized in five main tasks (from Task 3.1 to 3.5), each of which focusing on the definition and development of one or more components in the Horizontal Planes. **Task 3.1** aims at creating an efficient **Permissioned Blockchain** based on open-source alternatives such as Hyperledger Fabric, Quorum or Hyperledger Besu that will in turn support other ARCADIAN-IoT components such as identity management and reputation components. **Task 3.2** focuses on the development of **Hardened Encryption** mechanisms to protect private data in resource constrained devices. **Task 3.3** is devoted to creating **privacy preserving** technologies: (i) a dependable and privacy preserving classifier based on **Federated AI** algorithms which will also guarantee the source and data integrity, and (ii) a **Self-aware Data Privacy** component that will enhance the way data privacy is managed in IoT contexts. A novel IoT-specific **Cyber Threat Intelligence** system based on the MISP (Malware Information Sharing Platform) toolset which will provide privacy preserving data sharing capability and IoT-specific Indicators of Compromises is being developed in **Task 3.4**. Finally, **Task 3.5** aims to define and implement the monitoring and recovering functionalities which will be provided by the following components: (i) a **flow monitoring** agent, enhancement of existing Network Intrusion Detection Systems able to detect known malicious Distributed Denial of Service (DDoS) along the entire IoT infrastructure, (ii) a **Behaviour Monitoring** component able to detect anomalous behaviours that occurs on IoT devices, (iii) a **Network Self-healing**, (iv) **IoT Network Self-protection**, and (v) **IoT Device Self-protection** capabilities to mitigate and recover from cyberattacks against IoT networks and devices are developed in **Task 3.5**.

TABLE OF CONTENTS

EXECUTIVE SUMMARY **5**
TABLE OF CONTENTS **6**
LIST OF FIGURES **8**
LIST OF TABLES **9**
ABBREVIATIONS **10**
1 INTRODUCTION **13**
1.1 Objectives 14
2 PRIVACY PLANE **16**
2.1 Self-aware Data Privacy 16
2.1.1 Overview 16
2.1.2 Technology research 17
2.1.3 Future work 19
2.1.4 Current resources 19
2.2 Federated AI 20
2.2.1 Overview 20
2.2.2 Technology research 21
2.2.3 Future work 23
2.2.4 Current resources 23
3 SECURITY PLANE **24**
3.1 Network Flow Monitoring 24
3.1.1 Overview 24
3.1.2 Technology research 25
3.1.3 Future work 27
3.1.4 Current resources 28
3.2 Behaviour Monitoring 29
3.2.1 Overview 29
3.2.2 Technology research 30
3.2.3 Future work 34
3.2.4 Current resources 35
3.3 Cyber Threat Intelligence 36
3.3.1 Overview 36
3.3.2 Technology research 38
3.3.3 Future work 39
3.3.4 Current resources 40
3.4 Network Self-protection 41
3.4.1 Overview 41

3.4.2 Technology research 42
3.4.3 Future work....44
3.4.4 Current resources....45
3.5 IoT Device Self-protection....46
3.5.1 Overview 46
3.5.2 Technology research 47
3.5.3 Future work....51
3.5.4 Current resources....51
3.6 Network Self-healing....52
3.6.1 Overview 52
3.6.2 Technology research 53
3.6.3 Future work....55
3.6.4 Current resources....55
4 COMMON PLANE....56
4.1 Hardened Encryption....56
4.1.1 Overview 56
4.1.2 Technology research 57
4.1.3 Future work....63
4.1.4 Current resources....64
4.2 Permissioned Blockchain....66
4.2.1 Overview 66
4.2.2 Technology research 67
4.2.3 Future work....71
4.2.4 Current resources....71
5 CONCLUSIONS....72
REFERENCES....73

LIST OF FIGURES

Figure 1. ARCADIAN-IoT Conceptual representation highlighting Horizontal Planes. 13

Figure 2. Self-aware Data Privacy logical architecture. 16

Figure 3. Draft Data Model for policy specification. 18

Figure 4. Overview of Data rebalancing approach. 21

Figure 5. Overall schema representing the internal design of Network Flow Monitoring component deployed on the Cloud Network. 25

Figure 6. 5G multi-stakeholder network segments. 26

Figure 7. Nested encapsulation example 5G frame between Edge and Core networks. 26

Figure 8. Comparison between SIDS and AIDS approaches. 30

Figure 9. General Architecture of the centralized FL model. 33

Figure 10. General Architecture of the decentralized FL model. 33

Figure 11. The architecture of CTI. 36

Figure 12. Network Self-protection architecture. 42

Figure 13. IoT Device Self-protection. 46

Figure 14. MAPE-K architecture flow representation. 48

Figure 15. Risk Levels – C. Raibulet et al. 49

Figure 16. eSIM at device self-protection. 50

Figure 17. Architecture of the Network Self-healing component. 52

Figure 18. Architecture design of the UWS Resource Inventory Agent. 54

Figure 19. Attribute-Based Encryption with multiple authorities. 58

Figure 20. eSIM security abilities - GSMA IoT SAFE specifications. 59

Figure 21. eUICC structure comprising the IoT SAFE applet. 59

Figure 22. GSMA IoT SAFE components and their relation. 61

Figure 23. Attribute-Based Encryption architecture with integrated hardware RoT and blockchain support. 63

Figure 24. Example of a blockchain network with three peer nodes. 67

LIST OF TABLES

Table 1. Mapping the components to WP tasks. 14

Table 2. Example of a system call table (Linus 2.6.38). 31

Table 3. Types of attacks in ADFA-LD. 32

Table 4. Confusion Matrix. 34

Table 5. Selected methods for the first prototype of ARCADIAN-IoT security eSIM applet. 61

ABBREVIATIONS

3PP	3rd Party Provider
5G	5th Generation
5GS	5G System
ABE	Attribute Based Encryption
ADFA-LD	Australian Defence Force Academy Linux Dataset
AI	Artificial Intelligence
AIDS	Anomaly IDS
API	Application Programming Interface
CI/CD	Continuous Integration (CI) and Continuous Delivery
CP-ABE	Ciphertext-Policy ABE
CSIRT	Computer Security Incident Response Team
CTI	Cyber Threat Intelligence
dApp	decentralized Application
DARPA	Defense Advanced Research Projects Agency
DCSP	Data Centre Service Provider
DDoS	Distributed Denial of Service
DID	Decentralized Identifier
DLT	Distributed Ledger Technology
DSC	Datapath Security Controller
DTLS	Datagram Transport Layer Security
ECC	Elliptic Curve Cryptography
ECDHE	Elliptic-curve Diffie–Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
eSIM	embedded SIM
eUICC	embedded Universal Integrated Circuit Card
FE	Functional Encryption
FL	Federated Learning
FTP	File Transfer Protocol
GENEVE	Generic Network Virtualization Encapsulation
GRE	Generic Routing Encapsulation
GSMA	Global System for Mobile Communications Association
GSMA-SAS	GSMA's Security Accreditation Scheme
GTP	GPRS (General Packet Radio Service) Tunnelling Protocol
GUI	Graphical User Interface
HE	Hardened Encryption
HIDS	Host-based IDS
ID	Identifier
IDS	Intrusion Detection System
IID	Independent and Identically Distributed
IMSI	International Mobile Subscriber Identity

IoC	Indicator of Compromise
IoT	Internet of Things
IoT SAFE	IoT SIM Applet For Secure End-2-End Communication
IP	Internet Protocol
IPR	Intellectual Property Rights
IPS	Intrusion Prevention System
IT	Information Technology
JSON	JavaScript Object Notation
KDD	Knowledge Discovery and Data Mining
KPI	Key Performance Indicator
LTE	Long-Term Evolution
M2M	Machine to Machine
MAC	Media access control
MISP	Malware Information Sharing Platform
ML	Machine Learning
MPLS	Multiprotocol Label Switching
MVNO	Mobile Virtual Network Operator
NAA	Network Access Application
NB-IoT	Narrowband IoT
NBI	North Bound Interface
NFM	Network Flow Monitoring
NIDS	Network-based IDS
NIST	National Institute of Standards and Technology
OPA	Open Policy Agent
OS	Operating System
OVS	OpenVSwitch
PAP	Policy Administration Point
PCA	Protection Control Agent
PD	Protection Decider
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PHP	Hypertext Pre-processor
PoA	Proof of Authority
PPP	Public Private Partnership
REST	Representational State Transfer
RIA	Resource Inventory Agent
ROC	Receiver Operating Characteristic
RoT	Root of Trust
RSA	Rivest, Shamir, and Adelman encryption technology
RPC	Remote Procedure Call
SDN	Software Defined Network
SFMA	Security Flow Monitoring Agent

SHA	Secure Hash Algorithm
SHDM	Self-Healing Decision Manager
SIDS	Signature IDS
SIM	Subscriber Identity Module
SMOTE	Synthetic Minority Oversampling Technique
SP	Service Provider
SSH	Secure Shell
STIX	Structured Threat Information eXpression
SVM	Support Vector Machine
TAXII	Trusted Automated eXchange of Intelligence Information
TCP	Transmission Control Protocol
TEID	Tunnel Endpoint Identification
TLS	Transport Layer Security
TPR	True Positive Rate
UDP	User Datagram Protocol
UE	User Equipment
UICC	Universal Integrated Circuit Card
UNM	University of New Mexico
VISP	Virtualisation Infrastructure Service Provider
VLAN	Virtual Local Area Network
VNID	Virtual Identifier
VxLAN	Virtual Extensible Local Area Network
WP	Work Package
XDP	eXpress Data Path
XML	eXtensible Markup Language

1 INTRODUCTION

The ARCADIAN-IoT project aims to develop a cyber security framework relying on a novel approach to manage and coordinate, in an integrated way, identity, trust, privacy, security, and recovery in IoT systems. The proposed approach organizes the multiple cyber security functionalities offered by the framework into several planes combined together in an optimized way to support the end-to-end services. In particular, the framework includes three Vertical Planes devoted to identity, trust, and recovery management, and three Horizontal Planes supporting the Vertical Planes by managing privacy of data, monitoring security of entities, and providing Permissioned Blockchain and Hardened Encryption technologies (see Figure 1).

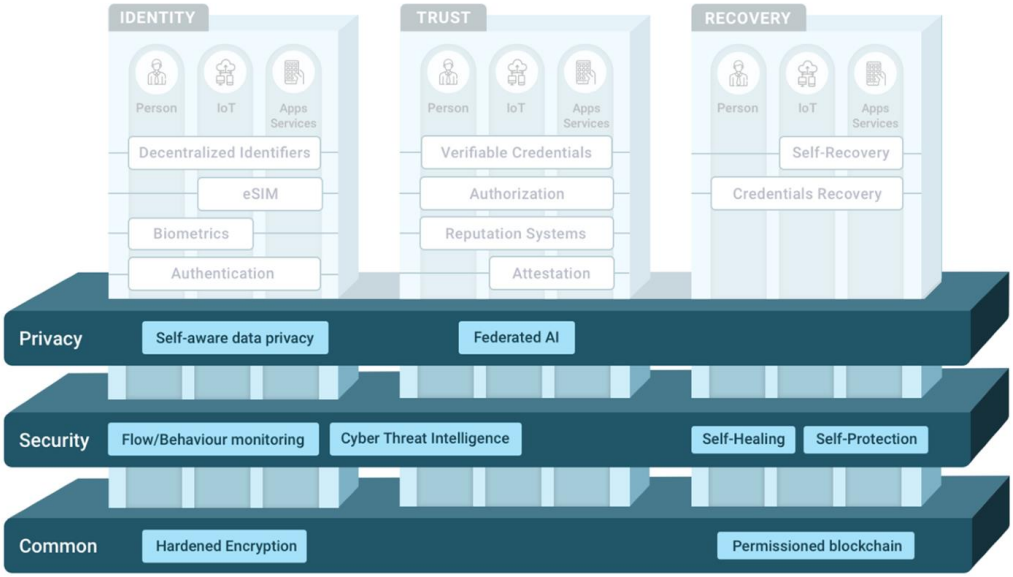

Figure 1. ARCADIAN-IoT Conceptual representation highlighting Horizontal Planes.

Work Package 3 (WP3) in the ARCADIAN-IoT project is dedicated to the design and technological development of the functionalities that are mapped into the Horizontal Planes for each selected use case. It is organized in five tasks, each one focusing on one or more components. The research activity in WP3 is being conducted from October 2021 to October 2023 and this deliverable (D3.1) details the initial research activities and preliminary findings obtained within WP3 until April 2022 (seven months). The next deliverable D3.2 will provide an update on the research achievements obtained till December 2022, while the final description of the developed components in WP3 together with the final results will be provided in deliverable D3.3 in October 2023. This document also includes the links to the open access source code that has been produced during the reporting period and not subjected to any Intellectual Property Rights (IPR) restrictions.

The Horizontal Planes of the ARCADIAN-IoT framework (consisting of ten main components) are organized as follows:

- The **Privacy Plane**, which aims to provide functionalities for the privacy-preserving management of confidential or sensitive data involving persons' entities, includes the (i) Self-aware Data Privacy and (ii) Federated Artificial Intelligence (Federated AI) components.
- The **Security Plane** contains all the cyber security features required for the monitoring, prevention, management, and recovery; it comprises the (i) Network Flow Monitoring, (ii) Behaviour Monitoring, (iii) Cyber Threat Intelligence, (iv) Network Self-protection, (v) IoT Device Self-protection, and (vi) Network Self-healing components.
- The **Common Plane** includes the two components that provide common functionalities to

the Vertical Planes, i.e., (i) the Hardened Encryption and (ii) Permissioned Blockchain.

Table 1 shows how the components in the Horizontal Planes are grouped per plane and in which task of the project are developed.

Table 1. Mapping the components to WP tasks.

Plane	Component	Task
Privacy Plane	Self-aware Data Privacy	3.3
	Federated AI	3.3
Security Plane	Network Flow Monitoring	3.5
	Behaviour Monitoring	3.5
	Cyber Threat Intelligence	3.4
	Network Self-protection	3.5
	IoT Device Self-protection	3.5
	Network Self-healing	3.5
Common Plane	Hardened Encryption	3.2
	Permissioned Blockchain	3.1

This deliverable is organized in three main sections mapping the three Horizontal Planes. Every section is split into multiple subsections, each of which describing the research findings for a specific component in the plane. The description of a component reports: (i) an overview of the developed component, including its structure in subcomponents, together with a recall of related requirements and Key Performance Indicators (KPIs); (ii) a summary of the research activities and findings related to the component for the current reporting period, (iii) the planned future work, and (iv) the references to the resources currently available that are shared by the partners, whenever possible.

1.1 Objectives

WP3 aims at contributing to the achievements of six main objectives in the ARCADIAN-IoT projects. Each objective comes with individual Key Performance Indicators (KPIs). The objectives are as follows:

- **To create a decentralized framework for IoT systems - ARCADIAN-IoT framework.**
  - *The Key Performance Indicator of this objective comprises the achievement of all the ensuing objectives and their KPIs.*
- **Enable distributed security and trust in management of persons' identification.**
  - *Facilitate deployment of blockchain technologies by non-cyber security experts in cyber security training sessions with, at least 20 participants.*
- **Provide distributed and autonomous models for trust, security, and privacy – enablers of a Chain of Trust.**
  - *Enable Federated AI mechanisms for, at least three, heterogeneous devices and entities*
  - *Enhance robustness of AI models for trust and security management by a factor of 30% in real scenarios*
  - *Enable detection of anomalous behaviour with accuracy of 90%.*

- **Provide a Hardened Encryption with recovery ability.**
  - *Provide at least three encryption mechanisms with low overhead*
  - *Enable efficient encryption with Root of Trust (RoT) information*
  - *Support selective recovery ability in encryption mechanisms: who and what can be recovered.*
- **Self and coordinated healing with reduced human intervention.**
  - *Recovery, at least 95% of the system functionalities prior to anomalous behaviour*
  - *Support coordination of recovery to pre-defined trust levels*
  - *Reduce human intervention to the strictly required, in healing and recovery procedures.*
- **Enable proactive information sharing for trustable Cyber Threat Intelligence and IoT Security Observatory.**
  - *Promote sharing of IoT threat data in EU, respecting privacy, and data regulations*
  - *Enable a novel automated and privacy-preserved CTI approach exploiting the European MISP platform (MISP4IoT).*

2 PRIVACY PLANE

The ARCADIAN-IoT provides functionalities that enable the privacy management of confidential and sensitive data. The Privacy Plane aims to provide a comprehensive and interoperable privacy management toolset regardless of the IoT system and fully compliant with European data privacy regulations. This plane includes two main components: the **Self-aware Data Privacy** (Subsection 2.1) and the **Federated AI** (Subsection 2.2) components. The research activity related to the Privacy Plane and described in this Section is part of **Task 3.3** (Enablers of privacy preserving for AI and persons) in Work Package 3 (**WP3**).

2.1 Self-aware Data Privacy

This research activity relates to the design and development of the Self-aware Data Privacy component which is part of **Task 3.3** (Enablers of privacy preserving for AI and persons) in **WP3**.

2.1.1 Overview

2.1.1.1 Description

Within the scope of the ARCADIAN-IoT project, Martel is developing a component to empower the users to better control privacy of their data, in particular by allowing the definition of user-defined privacy policies for data, and by crowdsourcing policies specified on similar data. The Self-Aware Data Privacy component includes two main modules: a **policy management** module which enforces data privacy via the definition of privacy policies – in the context of this component policies relates specifically to attributed-based data encryption and/or anonymization -, and a **recommender** module which, by assessing policy similarity, suggests privacy policies.

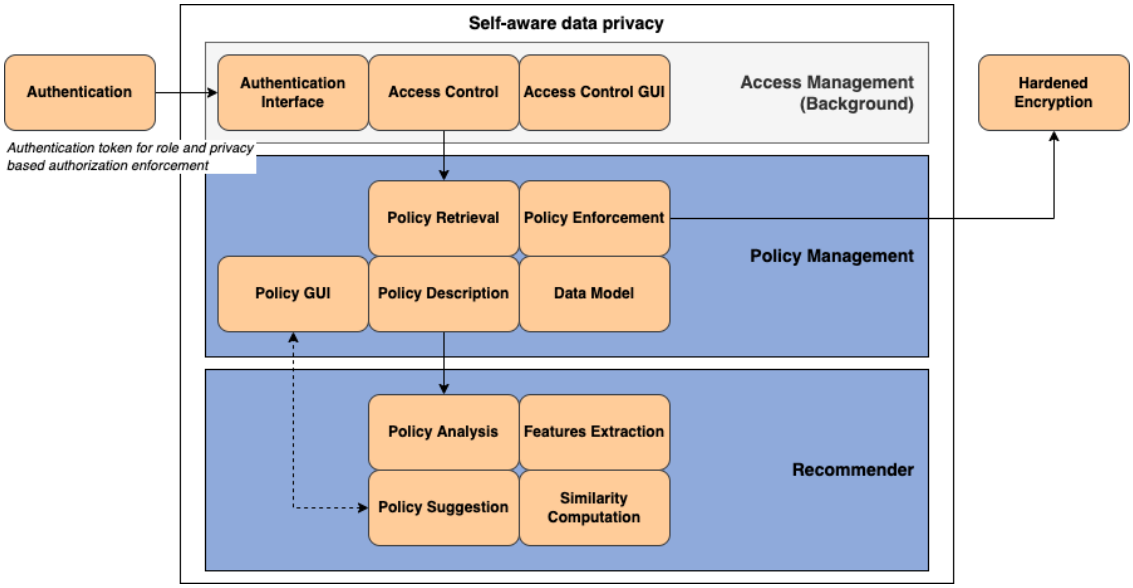

Figure 2. Self-aware Data Privacy logical architecture.

In Figure 2 we illustrate the logical architecture of the Self-aware Data Privacy component and its dependences with other internal and external components. The “Access Management” module is a background that Martel is employing to interface the Self-aware Data Privacy component with the Authentication component provided by Truphone (Deliverable D4.1 [1]). The integration with this background by Martel will also enable the exploitation of the Self-aware Data Privacy as a standalone component without requiring the full ARCADIAN-IoT software stack. The module is complemented by a Graphical User Interface (GUI) that allows end users to define access policies for the Self-aware data policy component. The core parts of the Self-aware Data Privacy are instead the “Policy Management” module and the “Recommender” one. The “Policy Management”

is itself organized into submodules that will allow: (i) end-users to specify custom policy to protect data or their attributes, potentially via a GUI ("Policy Description") following a "Data Model;" (ii) to retrieve data according to the specified policies ("Policy Retrieval"); and (iii) to enforce the policies by applying encryption/decryption or anonymisation/pseudonymisation to the data ("Policy Enforcement") with the help of the component develop by XLAB to provide attribute based encryption (cf. Subsection 4.1). The "Recommender" module will leverage the "Policy Management" one to analyse the already issued policies and, based on the extracted features (semantic, syntactical, etc) compute their similarity based on the data protected, and suggest the user the best policy to protect her/his data.

2.1.1.2 Requirements

A recall of the high-level requirements that have been previously defined and provided in deliverable D2.4 [2] is given below.

Requirement 2.1.1 – User defined policies: An authorized user can access the system and specify for a given data source or data property the security policies that allows to protect the data either when entering into the system or exiting – or both.

Requirement 2.1.2 – Policies validation: Security policies can be specified in a machine-readable format (e.g., JSON) and validated against a schema interpreter to assess their validity and applicability.

Requirement 2.1.3 – Data secured: Data sources or data attribute scan be secured by a given methodology: anonymisation, pseudo-anonymisation, encryption, and advanced encryption (hardened) based on attribute-based encryption.

Requirement 2.1.4 – Recommender: The system is able to recognise similarity between new data and existing data by key, attributes and/or semantic; This result is eventually displayed to users for facilitating the issuing of security policies.

2.1.1.3 Objectives and KPIs

With the aim of fulfilling the project's objectives, the main subcomponents of the Self-aware Data Privacy component will be tested through these KPIs:

- The ability to model policy for at least 5 encryption/anonymisation.
- Precision/recall combined in the F1 score for the recommender algorithms (≥80%).
- Usability of the policing issuer (Likert scale, 90% of positive feedback).

2.1.2 Technology research

2.1.2.1 Technical findings and achievements

The current work has focused on defining the architecture and a set of subcomponents for the Self-aware Data Privacy to achieve the goal of allowing data owner and managers to define policies for protecting those data and enforce their protection.

The architecture which we briefly described in Section 2.1.1.1 implements three state-of-the-art reference components (as defined in the OASIS XACML standard architecture[1]) through different open-source tools.

- **Policy Decision Point (PDP)**: is based on the open-source, general-purpose policy

[1] https://www.oasis-open.org/committees/xacml/

engine Open Policy Agent (OPA[2]), which decouples policy decision-making from policy enforcement, and it is designed to be employed in several deployment environments, from microservices to Kubernetes, from CI/CD pipelines to API gateways. OPA generates policy decisions by evaluating a query input (can be any structured data, e.g., JSON) against policies and data. The query input is informally identified as “Policy Retrieval” in Figure 2 and the interface between OPA and the “Authentication” API has been one of the focuses of this first part of the project.

- **Policy Administration Point (PAP)**: is a custom API currently under development in ARCADIAN-IoT and that will be released as open source under a permissive license schema (e.g., APACHE 2). This module corresponds to the “Policy Description” in Figure 2 and its main purpose is to provide to OPA the knowledge based upon which it will evaluate which technique (encryption/decryption, anonymisation) will be applied to protect data. The API follows the Data Model (cf. Figure 3) drafted up to know and that may further be refined to accommodate the specific needs of the Domains (WP5). At a later stage, the API will be complemented by a GUI to facilitate the management of the policies by the end users.

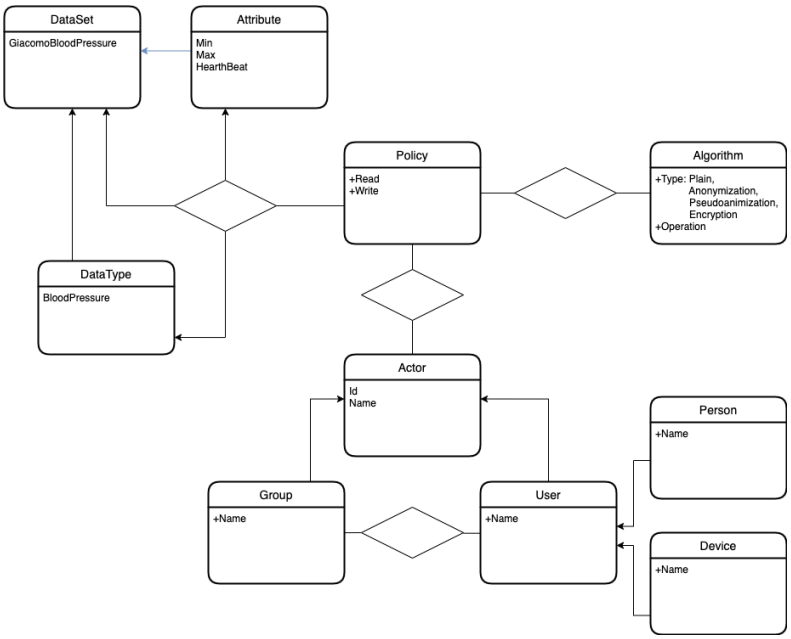

Figure 3. Draft Data Model for policy specification.

- **Policy Enforcement Point (PEP)**: will leverage envoy proxy[3] and provide a plugin for it to interface with PDP and PAP to unsure data is protected/accessed according to the policies specified. This component will leverage the encryption/decryption Libraries or API provided by XLAB (Section 4.1) to implement the security policies.

As regards the Recommender module, at the being we conducted a state-of-the-art study of the existing algorithms to assess structured document similarity like JSON schema by attributes, by keys, by semantic, etc. and state-of-the-art study of recommender system algorithms, which we plan to implement in the second part of the development cycle.

2.1.2.2 Evaluation approach

The evaluation will be done module by module in an isolated approach within Martel, then an

[2] https://www.openpolicyagent.org
[3] https://www.envoyproxy.io/docs/envoy/latest

integration test will be run along the whole chain, from the authentication to the policy enforcement to the data. In this stage fabricated data will be employed to allow for a smooth testing. Once the component will be robust enough, it will be deployed as integrated within the Domains with real (or close-to-real) data.

2.1.3 Future work

The future work will focus on finalising the API for policy definitions and its corresponding GUI, then to link the API to the PDP. The next step will be the implementation and testing of the PEP, i.e., the application of the algorithms for encryption/decryption or anonymisation/ pseudonymisation of the data. A first prototype will be released in the scope of deliverable D3.2, while the fully integrated system will be released at the end of the development cycle in deliverable D3.3. In parallel to this, we will design the architecture of the Recommender system based on the refined Data Model and curate its implementation, which will be released and documented in deliverable D3.3.

2.1.4 Current resources

The code of the component for the Access Control, can be found on GitHub, under the codename of “Anubis” and with an open-source license.[4]

[4] https://github.com/orchestracities/anubis

2.2 Federated AI

The research activity related to the design and development of the Federated AI component is part of **Task 3.3** (Enablers of privacy preserving for AI and persons) in **WP3**.

2.2.1 Overview

2.2.1.1 Description

Within the scope of the ARCADIAN-IoT project, Research Institutes of Sweden (RISE) is building dependable and privacy preserving Federated Learning (FL) capabilities which will be deployed in machine learning (ML)-based components (e.g., CTI and Behaviour Monitoring components). The proposed solution will provide both source integrity (the guarantee that no malicious participant is involved in the process) and data integrity (privacy of raw data and local model updates is preserved). In addition, RISE is investigating the problem of having statistical and systematic heterogeneity of data in IoT environments, which usually determines an inaccurate and vulnerable training process.

Federated AI component will be developed as integrated module within those ARCADIAN-IoT components that provide ML models as their functionalities (e.g., CTI and Behaviour Monitoring component). It includes two subcomponents: data rebalancer and model resizing and sharing. Data rebalancer will provide a way to rebalance and fit non-Independent and Identically Distributed (non-IID) data to the framework; Model resizing and sharing will implement a communication-efficient and robust framework for model aggregation while preserving source integrity. This subcomponent accelerates and protects the local model from being attacked by adversarial attacks from malicious entities. Data integrity will be provided by (i) the data rebalancer which will only share generated synthetic data and (ii) the model resizing and sharing subcomponent which will use standard FL paradigm to share only processed ML models.

2.2.1.2 Requirements

A recall of the high-level requirements that have been previously defined and provided in deliverable D2.4 [2] is given below.

Requirement 2.2.1 – Malicious behaviours from sharing entities to be detected: When an entity collaborating in the FL setup misbehaves during the learning process, it should be detected in order to invalidate the output.

Requirement 2.2.2 – Local ML model to be lightweight: The models that are generated locally by the CTI should be lightweight, as this will make the FL more efficient in the federated model computation.

Requirement 2.2.3 – At least three heterogeneous devices/entities: The Federated AI mechanisms should support the training among at least three heterogeneous devices and entities.

2.2.1.3 Objectives and KPIs

The main objective is to provide a dependable privacy preserving classifier based on FL which (i) incorporates classical data balancing techniques traditionally used for dealing with imbalanced date in centralized ML, and (ii) ensures source and data integrity in the learning phase. With the aim of fulfilling the project's objectives, the Federated AI module will be evaluated against the following KPIs:

- Enable Federated AI mechanisms for at least three heterogeneous devices and entities.
- Provide detection of anomalous behaviour with accuracy of 90%.
- Enhance computational complexity of data rebalancer for FL by a factor of 20% in real

scenarios.

- Enhance robustness of AI models for trust and security management by a factor of 30% in real scenarios.

All these KPIs are related to the main project objective "*Provide distributed and autonomous models for trust, security and privacy – enablers of a Chain of Trust*" (as recalled in Introduction).

2.2.2 Technology research

2.2.2.1 Technical findings and achievements

During the current reporting period, RISE has focused on the definition and development of methods for rebalancing non-IID and imbalanced data. The research activity performed can be summarized as follow:

- Studied the issues related to the statistical heterogeneity (non-IID data) within Federated AI and IoT network, which usually causes a degradation of training the models.
- Explored and analysed three state-of-the-art data rebalancing methods.
- Designed and implemented a new potential solution for data rebalancing.

Statistical heterogeneity can arise from non-IID data collection in IoT networks, because of the FL setups, the number of the data points or data distribution may vary significantly across devices or clients, which would degrade the performance of the model. Imbalanced data is a common issue in real-world classification tasks. It refers to the problem that one class is heavily under-represented compared to the other class, in a two-class classification problem. It affects the performance of the classifier since prediction leans towards the majority class, and the minor class is usually wrongly classified. This situation is typical for many cyber security applications, including anomaly, attack, or fraud detection, where datasets are often quite imbalanced. The state-of-the-art data rebalancing techniques include: (i) oversampling [3], (ii) under-sampling [4], and (iii) Synthetic Minority Oversampling Technique (SMOTE) [5,6]. Oversampling and under-sampling are two classic techniques to address this issue. Nonetheless, the application of these techniques has not been explored in contexts for FL.

After understanding the challenges and situations of previous works, RISE has studied an adaptive data rebalancing technique, which can be used in peer-to-peer FL, and for non-IID data. Starting from the K-SMOTE [7], a variation of the original SMOTE for IoT settings, RISE has defined a new technique able to generate more complex synthetic points to share some of them with other participating clients. Figure 4 depicts how the proposed Data rebalancer subcomponent works.

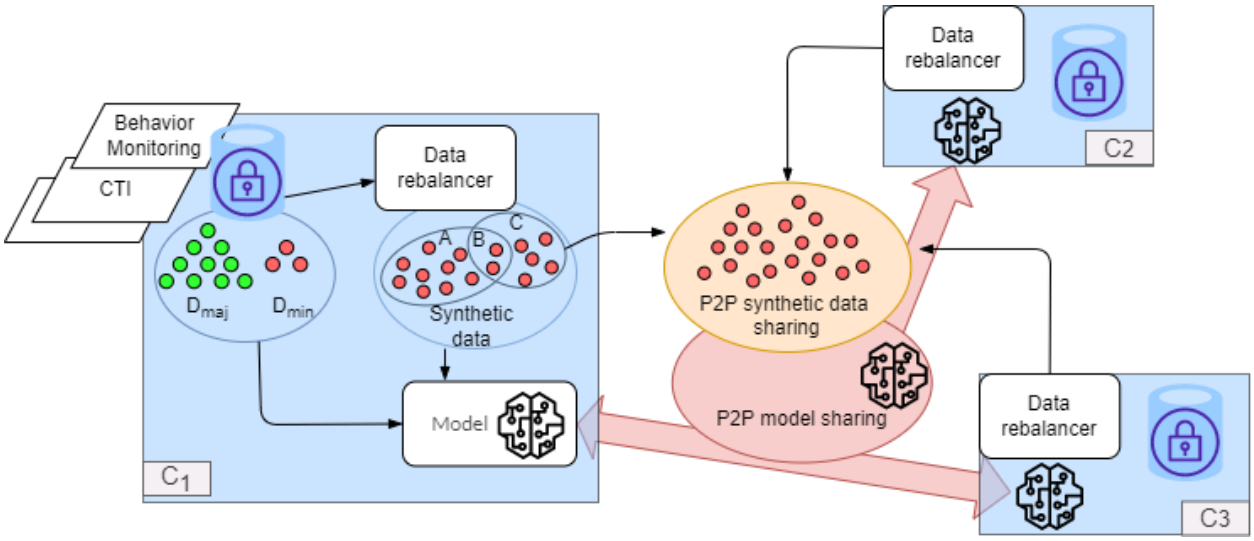

Figure 4. Overview of Data rebalancing approach.

Within an IoT network, the Data rebalancer can be deployed on the edge devices, referred to as clients in FL setups. The edge device can be, for example, a gateway that includes Behaviour Monitoring capabilities in order to detect any malicious activities. In our proposed solution, there is no central node required to participate to or orchestrate the training process. Overall, the entire process that involves the data balancer consists of three phases: (i) synthetic data generation, (ii) model optimization, and (iii) data and model sharing. In the first phase, the training data available at a client are rebalanced by an over-sampling technique (i.e., augmenting the number of data points from the minority class) which generates synthetic data points from the genuine data points in the minority class. The synthetic data points are randomly split into three subsets (A, B, and C). The synthetic data points in the set $(A \cup B)$[5] are merged with the genuine data points so as to obtain a balanced dataset which is fed into the local ML model stored by the client. Then, the ML model is trained with the merged balanced dataset. Finally, the resulting model and synthetic data points in the set $(B \cup C)$ are shared with a subset of the connected clients. It is important to note that the clients are not sharing directly training data points, but a mix of synthetic data points which are partially not used for training the local model.

Again, the cause of non-IID phenomenon is heterogeneity of various IoT devices, and this phenomenon degrades the model's performance in FL. It is bound to sacrifice some privacy to solve the problem of non-IID. Instead of having an auxiliary dataset maintained by the central node, the algorithm shares only some artificial data between some participants to help their local data rebalancing. The challenge is how to preserve the privacy of data when sharing the artificial data which is generated based on the raw data. SMOTE, as a state-of-the-art rebalancing method, has been used in many imbalanced data problems. It generates synthetic data by linear interpolation of samples in the minority class. However, it has the risk that the genuine data points are easy to be inferred when synthetic data is shared with other peers. To reduce the risk of privacy breaches, RISE defined and developed HSphere-SMOTE, an enhanced version of the SMOTE, suitable for FL sessions in IoT scenarios.

Algorithm 1: Algorithm of HSphere-SMOTE

Input :
D_{min}: set of the data points in the minority class
d_{min}: number of data points in D_{min}
$d_{sam}(< d_{min})$: number of data points to use for sampling
d_{syn}: number of synthetic data points to generate

Output:
S: Set of synthetic points

$R \leftarrow$ Set of d_{sam} randomly picked samples in the minority class
$d_{pe} = d_{syn}/d_{sam}$
$S = []$
for $x_i \in R$ **do**
  $N(\subseteq D_{min}) \leftarrow$ Set of nearest neighbors for x_i
  $S_{x_i} = []$
  $x_{c_i} = x_i + \frac{1}{|N|} \sum_{x_j \in N} (x_j - x_i) \times rand.beta(2,2)$
  $r = \max_{x_j \in N} \{(x_j - x_i)\}$
  $S_{x_i} \leftarrow$ Sample d_{pe} points from pdf $P_{HSphere}\{x, x_{c_i}, r\}$
  adjoin S_{x_i} to S
end

▷ $P_{HSphere}\{x, a, r\}$: probability density function over the volume of the hyper-sphere centered at a with radius r.
▷ $rand.beta(\alpha, \beta)$: random value drawn from the Beta distribution with two positive shape parameters α and β.

Algorithm 1 describes the whole process of HSphere-SMOTE in detail. First, the input to the

[5] The symbol ∪ is employed to denote the union of two sets.

algorithm consists of the dataset that includes labelled data from the minority class, and several hyper-parameters including the number of base data to re-sample and the number of synthetic points to be created. The output is the final set of synthetic points. HSphere-SMOTE first randomly picks d_{sam} base data points (line 1). For each base data point x_i, the algorithm samples d_{pe} synthetic points from a probability density function $P_{HSphere}$ with uniform distribution over the volume of a hypersphere centered at x_{c_i} and with radius equals to the maximum distance between x_{c_i} and any neighbours of x_i.

2.2.2.2 Evaluation approach

The proposed solutions for FL will be tested and evaluated in ICE Data center (Infrastructure and Cloud research & test Environment), a facility, owned by RISE, which provides Kubernetes cluster equipped with Nvidia-Gtx-2080ti GPU and 10 GB CUDA memory.

The performance of Data rebalancer will be evaluated both in traditional federated setting and peer-to-peer federated setting. It will be evaluated in four scenarios: (i) Globally balanced and IID data, (ii) Globally imbalanced and IID data, (iii) Globally imbalanced and mixed IID data, and (iv) Globally imbalanced and non-IID data. All four mentioned scenarios are considering the anomaly detection problem in IoT networks, which means it is a binary classification problem that include benign and malicious traffic.

The general evaluation metrics for classification, such as accuracy, are ambiguous and misleading in the case of problems involving imbalanced datasets, as the accuracy shows the ratio of correct classified among the whole dataset, which it easily biases towards to the majority class. We therefore choose False Negative Rate, False Positive Rate, Recall value when the precision reaches 75%, and Precision value when the Recall reaches 75% for each local model. Additionally, we will analyse the computational and the communication costs. The results will be compared against those obtained by the mentioned state-of-the-art solutions, including the most recent K-SMOTE algorithm.

2.2.3 Future work

Future work will focus on the following: (i) Performance evaluation on real-world datasets for rebalancer in federated setting for IoT scenarios, (ii) Definition of a method for model resizing and sharing with communication efficiency and robustness for IoT scenarios, (iii) Development of the method for model resizing and sharing and performance evaluation, (iv) Integration and deployment of the FL component and its subcomponents into the Behaviour Monitoring and CTI components. A first prototype will be prepared on time for deliverable D3.2, while the fully integrated system will be available at the end of the development cycle and provided in deliverable D3.3.

2.2.4 Current resources

Due to the early stage of development, there are no resources currently available to public. The source code of the data rebalancer will be released as open source once all the evaluations are completed. The same will apply to other code produced within this task.

3 SECURITY PLANE

The Security Plane of the ARCADIAN-IoT framework includes a set of tools dedicated to the cyber security incident detection, response, management, and recovery for IoT systems. The plane consists of a total of six components: (i) **Network Flow Monitoring** (focusing on the network), (ii) **Behaviour Monitoring** (focusing on devices), (iii) **Cyber Threat Intelligence**, (iv) **Network Self-healing**, (v) **IoT Device Self-protection**, and (vi) **Network Self-protection**.

The research activity related to the Security Horizontal Layer is part of **Task 3.4** (Cyber Threat Intelligence for IoT systems) and **Task 3.5** (Self-healing and self-protection for IoT systems) of **WP3**.

3.1 Network Flow Monitoring

The design and development of the Network Flow Monitoring component is part of **Task 3.5** (Self-healing and self-protection for IoT systems) in **WP3**.

3.1.1 Overview

3.1.1.1 Description

Within the scope of the ARCADIAN-IoT project, University of West of Scotland (UWS) is developing the Network Flow Monitoring (NFM) component that will act as an enhancement of existing Network Intrusion Detection Systems (NIDS), such as Snort,[6] to achieve the detection of known malicious Distributed Denial of Service (DDoS) along the entire infrastructure of the 5G network. As shown in Figure 5, this will be achieved by the contribution of two different subcomponents: (i) an NIDS with an updated set of rules for known DDoS attacks, and (ii) the Security Flow Monitoring Agent (SFMA) that will act as a wrapper for the NIDS alert and will provide fine data information about the malicious flow detected. This component will provide support not only for traditional IP networks, but also for the overlay networks currently used in cloud infrastructures employing overlay/encapsulation protocols such as Virtual Extensible LAN (VxLAN), Generic Routing Encapsulation (GRE), Generic Network Virtualization Encapsulation (GENEVE), those currently used in enterprise infrastructures such as VLAN, and those currently used in cellular and IoT mobile operator networks such as GTP used in LTE-M and NB-IoT.

[6] https://www.snort.org/

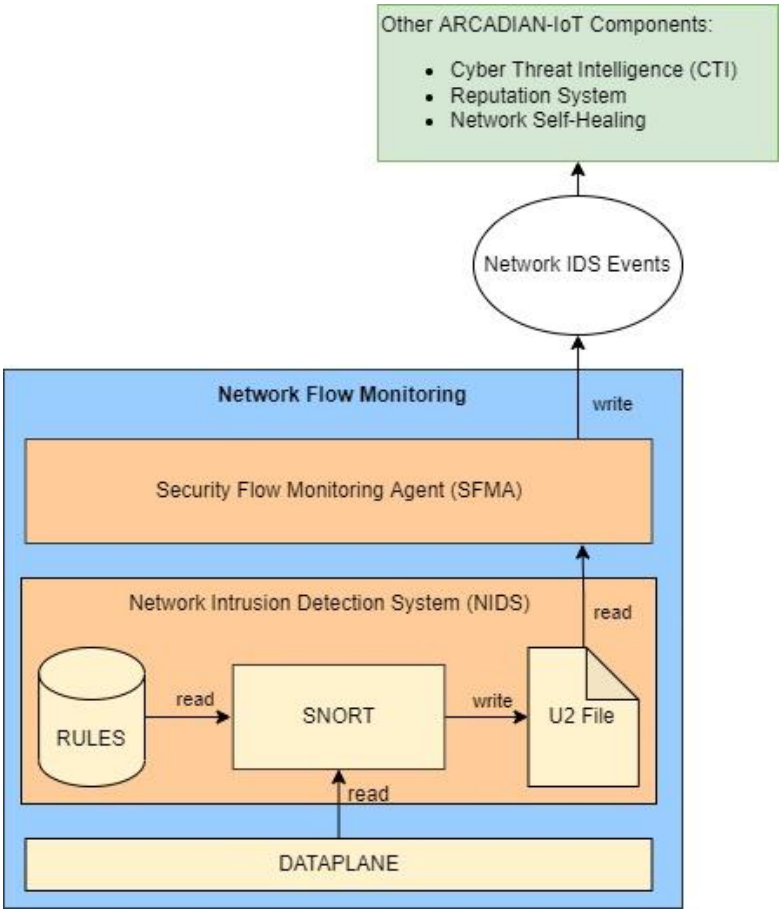

Figure 5. Overall schema representing the internal design of Network Flow Monitoring component deployed on the Cloud Network.

3.1.1.2 Requirements

A recall of the high-level requirement that has been previously defined and provided in deliverable D2.4 [2] is given below.

Requirement 3.1.1 – IoT Network Detection: The flow monitoring component will have the capabilities to perform the detection of DDoS attacks in any network segment of the IoT infrastructure, triggering the associated alert.

3.1.1.3 Objectives and KPIs

The aim of the Network Flow Monitoring component is to provide network intrusion detection capabilities for overlay networks into the ARCADIAN-IoT framework. With the aim of fulfilling the project's objectives, the following KPI will be used to assess and validate the performance of the Network Flow Monitoring component:

- Reduce human intervention to the strictly required, in healing and recovery procedures.

This KPI is related to the main project objective "*Self and coordinated healing with reduced human intervention*" (as recalled in Introduction). Thus, this component will further contribute to mitigate and reduce the number of cyber incidents involving IoT networks.

3.1.2 Technology research

3.1.2.1 Technical findings and achievements

The 5GS (5G System) architecture presents several different stakeholders that are involved in

the provisioning of 5G network resources, as presented in the View on 5G Architecture by the 5G PPP (5G Public Private Partnership) [8]. A key role in the provision of 5G services is that of the Service Provider (SP), which interacts directly with service customers and obtains and orchestrates resources from Network Operators, VISPs (Virtualisation Infrastructure Service Providers) and DCSPs (Data Centre Service Providers), collectively referred to as infrastructure providers. Figure 6 presents such a 5G scenario where there are physical resources that are shared by means of a virtual layer and where the main architectural elements of the 5G architecture are depicted. When a User Equipment (UE) is connected to an antenna, which belongs to operator A, this user is identified in the 5G networks by their TEID (tunnel endpoint identification) of their associated GTP tunnels. Also, some architectural elements of the 5G network are associated to a given tenant/operator, which is identified with its unique virtual ID (VNID) given by the VxLAN encapsulation. It means that each packet sent from one user to another user allocated in a different antenna must be encapsulated at least twice in the network segment between the edge and the core of the network. In a first stage VxLAN isolates the tenant traffic, and in a second stage, GTP provides user mobility. An example of this nested encapsulation can be found in Figure 7.

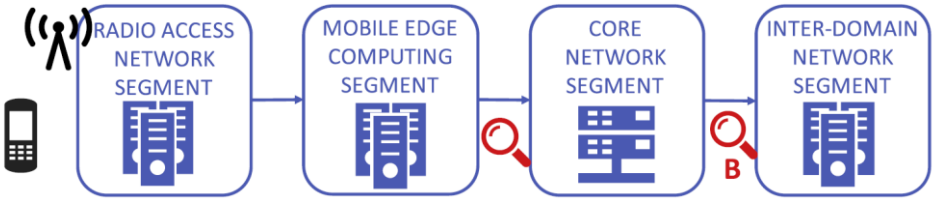

Figure 6. 5G multi-stakeholder network segments.

Figure 7. Nested encapsulation example 5G frame between Edge and Core networks.

Traditional signature-based NIDS such as Snort are mainly designed to provide detection capabilities to traditional IP networks. Thus, they do not provide support to detect transversal overlay networks being encapsulated over such IP networks, no IDS support VxLAN encapsulation which is needed for the tenant-isolation of traffic allowing L2 overlay networks. Also, even the most advanced IDS published up to date does not provide any support for double encapsulated traffic (nested encapsulation) which is exactly the main requirement imposed by the 5G multi-tenant architectures. This lack of support for transversal detection and nested encapsulation makes traditional IDS tools unsuitable for the new network traffic patterns imposed by 5G architecture. As a novel capability, the NIDS should be able to detect simultaneously attacks being addressed over a 5G user, a tenant, or the entire infrastructure. The proposed architecture for the NFM component makes the following contributions and achievements to IoT 5G Network security:

- Diverse types of traffic tagging and encapsulation protocols are supported such as VxLAN, GTP, GRE, and tagging protocols such as MPLS and VLAN allowing the usage of the NIDS in the edge and other network segments, unlike traditional NIDS solutions.
- This approach accomplishes one of the most important requirements of 5G Network protection systems, which is being tenant-aware, but also 5G user-aware.

After a deep study of the novelties and state of the art of the technologies that UWS is using within

NFM component (such as the NIDS and the proposed SFMA), the current status of the cited component is as follows: An analysis of different DDoS known attacks have been done, related to signature-based NIDS as explained in this section. It has been developed a tool that helps the researcher to deploy botnets with the main purpose of emulating known attacks over a 5G network infrastructure. These attacks can be launched from different numbers of attackers, representing one of the main characteristics that help studying the impact of volumetry of the attacks in the system. The tool developed is responsible of creating the IoT emulated devices with a DDoS tool installed in them.[7]

As mentioned in this section, there are some configuration and strategies that UWS will follow to retrieve a good deployment of the NIDS. The use of Unified2 protocol to report Snort logs is one of the most important, so this report can be binary stored, and this can help to improve the performance of the component.

Finally, UWS is currently focusing on the design of the SFMA subcomponent, which provides functionalities for retrieving nested encapsulation of the packets, as showed in Figure 7. Once this subcomponent is finally designed, the NFM design will be also finished.

3.1.2.2 Evaluation approach

Following the prototyping and as preliminary step towards the integration with the rest of the architectural components composing the different Horizontal Planes of the ARCADIAN-IoT framework, the overall behaviour of the NFM component will be empirically tested. The tests will be carried out in the cloud data centre at the UWS facilities and will be focused on two main aspects: functional validation and overall performance.

For validating the functionality of the component, it will be exposed to a set of scenarios, in a 5G system, where different known DDoS attacks will be launched. This will ensure that the effectiveness of the component is as expected, as the component is designed to trigger alerts in the moment that one of the network flows matches any of the rules stored into the Snort rules database.

On the other hand, the NFM component will be critically analysed in different areas to demonstrate its performance in different situations. In particular we will analyse:

- **Scalability –** Maximum Number of IoT devices traffic handled.
- **Bandwidth** – Maximum data rate that the component is able to deal with.
- **Delay** – Average time to process each alert; This involves the time consumed by Snort to match a rule and trigger the alert to the unified2 log file, the time consumed by the SFMA to analyse the alert and trigger the new one with the full information of the overlay networks to the following components.

3.1.3 Future work

Currently the UWS team is engaged in the development of a first functional prototype. The development of a Botnet IoT traffic emulator is the first step achieved in this prototype and follows the configuration of the NIDS and the development of a SFMA prototype. A first version of this prototype will be available during the second year of the project.

After prototyping and as a last step in the development of the component, the prototype will be empirically validated and evaluated (as described in previous Subsection 3.1.2.2). Hence, a last version of the prototype will be provided in deliverable D3.2 and an empirical validation and evaluation of the overall performance of this component will be reported in deliverable D3.3 with

[7] https://github.com/markus-go/bonesi

the subsequent integration in the overall ARCADIAN-IoT framework.

3.1.4 Current resources

Due to the application of IPR safeguarding measures, UWS has not the 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 Network Flow Monitoring component will be released for the ARCADIAN-IoT consortium.

3.2 Behaviour Monitoring

The design and development of the Behaviour Monitoring component is part of **Task 3.5** (Self-healing and self-protection for IoT systems) in **WP3**.

3.2.1 Overview

3.2.1.1 Description

In order to enhance the security of the IoT devices, the ARCADIAN-IoT project is set out to develop a component that aims to detect anomalous behaviour that occurs on device level. The component at hand is a host-based intrusion detection system (HIDS) specialized for IoT devices, which takes the task of examining incoming events that are specific to the host device (such as what applications are being used, what files are being accessed, permission changes, sequences of system calls and authentication attempts).

The component will be comprised by a set of subcomponents that aim to collect and classify the events with the use of lightweight ML models. The models are going to be updated via a FL scheme, which will ensure the preservation of privacy of the participating devices in the network.

The detection is to be performed in real time to make sure that everything is running normally and detect any sign of intrusion. The resulting output of this component will consist of a value that indicates if the event was classified as an intrusion or normal, if there was an intrusion detected, the component should also be able to classify the intrusion from the existing known types of attacks. The information will be packaged and forwarded to other ARCADIAN-IoT components, for instance, if there was an intrusion detected, the information should be received by the IoT Device Self-protection component, where its role is to protect devices against the incoming attacks.

3.2.1.2 Requirements

A recall of the high-level requirements that have been previously defined and provided in deliverable D2.4 [2] is given below.

Requirement 3.2.1 – User logs access: The Behaviour Monitoring component must have access to device's user logs in order to detect possible security issues.

Requirement 3.2.2 – Device permissions: The Behaviour Monitoring component must be aware of permissions granted to applications/services accessing the device.

Requirement 3.2.3 – Local model training capabilities: The ML models should be lightweight and be able to run on the device.

Requirement 3.2.4 – Response to anomalies: The Behaviour Monitoring component should be able to send an alarm, in real time, when anomalous behaviour is detected.

Requirement 3.2.5 – Secure Communication: The Behaviour Monitoring component communications between the devices and the central server should be secured (e.g., using encryption).

3.2.1.3 Objectives and KPIs

The main objective of the Behaviour Monitoring component is to detect any anomalous behaviour, in real time, on IoT devices based on device level events. The component will be evaluated against the following KPIs:

- Enable detection of anomalous behaviour with accuracy of 90%.
- Response time of the system when an intrusion occurs be not more than 30s.

- Enable local training in at least 2 different types of devices (e.g., smartphone, IoT GW).
- Use FL in the anomaly detection pipeline in at least 2 different types of devices (e.g., smartphone, IoT gateway).

All these KPIs are related to the main project objective "*Provide distributed and autonomous models for trust, security and privacy – enablers of a Chain of Trust*" (as recalled in Introduction).

3.2.2 Technology research

3.2.2.1 Technical findings and achievements

Host-Based Intrusion Detection System (HIDS) solutions are typically deployed across networks and systems and not directly on IoT devices. However, vulnerabilities in IoT devices are becoming more easily exploitable as the types of attacks are equally evolving. To this end, research is turning towards understanding trends within system data to be able to detect intrusions quicker and more efficiently in IoT devices. Cyber security becomes ever more necessary as more IoT devices flood the market. As such, the Behaviour Monitoring component will act as a lightweight HIDS that is capable of running directly on IoT devices.

HIDSs have been shown to be an effective means of protection on modern systems. Such systems are able to detect intrusions on an individual host by analysing the information that is available on the machine (e.g., system calls, network traffic, or others).

Literature shows that several methods for intrusion detection have been proposed. These can be categorized as either Signature Intrusion Detection System (SIDS), which can be described as a process where a unique pattern is established about a known threat so that the threat can be identified in the future, or as Anomaly Intrusion Detection System (AIDS), that focuses on detecting behaviour that is unknown to the system. Figure 8 depicts the main differences between the two methods.

	Advantages	Disadvantages
SIDS	• Very effective in detecting intrusions with a very low false positive rate • Promptly identifies the intrusions • Superior in detecting already known attacks • Simpler design	• Needs to be updated frequently with new signatures • If an existing known signature (intrusion) suffers from slight deviations, the system wouldn't be able to detect that signature. • Is not able to detect zero-day attacks. • Not suited to detect multistep attacks.
AIDS	• Can be used to detect unknown attacks • Could be used in order to create an intrusion signature	• Has a high false positive rate • Hard to build a normal profile for a dynamic computer system • Unclassified alerts • Needs initial training

Figure 8. Comparison between SIDS and AIDS approaches.

SIDSs are most commonly used for identifying threats that are already known. However, as expected, SIDSs show difficulty in detecting zero-day attacks because there is no matching signature in the database until the signature of the new attack is added.

On the other hand, an anomaly-based system has a constructed model of normal behaviour. AIDSs have drawn interest from a lot of cyber security researchers due to its capacity to overcome the limitations imposed by SIDS. The detection system indicates an attack when the observed

behaviour statistically deviates from the modelled normal behaviour.

The development of AIDS is composed of two phases: the training phase and the testing phase. In the training phase, normal traffic profiles are used to teach a model of normal behaviour. Then, in the testing phase, a new data set is used (preferably with data that has intrusion traces) to evaluate the system's capacity to generalize intrusions, even if the intrusions are zero-day attacks.

Let's consider a scenario where a device is being used on an open network and a third party intends to perform malicious actions on the said device. To detect these malicious actions, one way we can go about doing this is to search for specific characteristics of the attack. For instance, the attacker performs actions/commands on the system that are different from the way a typical user would. This can be used to create a detection system that will search for these deviations from the normal behaviour on the IoT device. Some examples of these anomalies are unusual read/write (or other system calls) behaviour or multiple login attempts in a certain timeframe to the machine.

A system call is a fundamental interface between a program on a machine and the operating system. The system call is a way for a user program to request an operation to be completed by the operating system. For example, a program may want to open a file and then copy it to a new directory. These tasks can be accomplished by following a sequence of system calls. As a result, programs running on the machine will produce sequences of system calls, also known as system call traces. The amount of possible system calls can be described as the alphabet based on the operating system. The size of the alphabet is generally large and depends on the system architecture, for example, the Linux 2.6.38 has a dictionary with a total amount of 324 distinct system calls (see Table 2).

Table 2. Example of a system call table (Linus 2.6.38).

Number	Name
0	sys_restart_syscall
1	sys_exit
2	sys_fork
3	sys_read
...	…
265	sys_clock_gettime
...	…
322	sys_timerfd_create
...	..

System calls (or syscalls) are a popular data source for HIDSs because they are a primary artifact of the OS kernel, and their collection imposes low performance overhead. Often, the unit of data used for detection is a system call trace, a sequence of all calls invoked by a single process in a given time window. System calls can be collected, for instance, with the *'strace'* Linux utility, although there are many other ways to collect this same information. Some common calls include *'open,' 'close,' 'read,' 'write,' 'wait,' 'exit,' 'mmap,'* among many others. The effective use of system calls within host-based anomaly detection was first proposed by Forrest et. al. [9] with the following advantages:

- System calls (root processes) are more advantageous than a typical user process because a system call will have greater access to the system's resources.
- An operating system will have a finite list of operations, which creates predictable system call sequences under normal system operations.

Many works on HIDS research focus on the analysis of system call traces. For instance, Haider et al. [10] proposed using different, but still inexpensive, statistical features on system call traces of the ADFA-LD dataset, with the same goal of fast performance of transforming data to features

without sacrificing accuracy of detection. Four features, namely, the least/most repeated and the minimum/maximum values in a trace, are used to represent a trace to detect attacks, and three supervised learning algorithms, Support Vector Machine (SVM) with linear and radial basis kernels and k-Nearest Neighbours, are used.

The processing of system calls for intrusion detection is typically based on payloads as a sequence of bytes as well as the operation of feature selection via overlying byte tuples, referred to as N-grams. We are then going to analyse the data exploring the notion of subsequences. N-grams have been used as models to understand sequence data as well as in predicting a sequence of events. An n-gram is a short arbitrary length, denoted by the n, subsequence of a larger sequence. In this context, we can take sub sequences of the system call traces of some length.

Many datasets are available as open source that tackle system call traces, one is the Australian Defence Force Academy Linux Dataset (ADFA-LD). This dataset was specifically designed to address limitations of previously collected datasets such as the DARPA, KDD, and UNM datasets. In particular, they captured system call traces on a server running a modern operating system (Linux) with realistic workloads (e.g., web browsing and word processing) and attack sequences generated via real vulnerabilities in commonly used software. For these reasons, the ADFA-LD dataset is often used for HIDS research, and previous work has demonstrated that this realism translates into a much more challenging learning task, suggesting that realistic datasets are vital for designing systems for practical deployment.

In Table 3 we can see the breakdown for the several types of attacks in the ADFA-LD dataset as well as the methods used to implement them. The small vulnerabilities were purposefully engineered into the system to be able to collect the data appropriately.

Table 3. Types of attacks in ADFA-LD.

Attack Name	**Method**
Brute force Password	FTP by Hydra
Brute force Password	SSH by Hydra
Add new superuser	Client-side malicious executable
Java Based Meterpreter	Tiki Wiki vulnerability
Linux Meterpreter	Client-side malicious executable
C100 Web shell	PHP Remote file Inclusion Vulnerability

This set of attacks represents what an average attacker may use in an attempt to take advantage of a system and compromise it. The set ranges from low-level, high-profile brute force attempts to vulnerabilities within services running on the machine. The use of these attacks will yield traces that use modern hacking methods.

The first two attack types represent a brute force password guessing attempt on the open FTP and SSH services. This attack is often a last resort of most attackers due to its large potential to be caught out on any Intrusion Detection Systems (IDSs). For the next attack, an executable can be used to create a new user with root privileges. The next two attacks involve Meterpreter - a command shell with enhanced functionality. The TikiWiki vulnerability allows to unknowingly upload a copy of Java Meterpreter. Once the program is uploaded it would initiate a reverse TCP connection to attack the system. Once the command shell is installed, it can have greater access to changing the configuration of the system, escalating privileges or attempting to access the shadow password file. The Linux Meterpreter can be uploaded using social engineering and behaves similarly to the Java Meterpreter. The main difference between the two is the implementation of the program itself, which could lead to different system call traces. Lastly, the

C100 Webshell is a sophisticated piece of PHP which allows for remote shell access.

Aside from the datasets related to attack types, the ADFA-LD contains system call “traces” of a system under a normal operation.

In Figure 9 and Figure 10, we represent the architectures for an end-to-end process for our approach, which will be developed in parallel to each other. The proposed architectures are: a centralized (cf. Figure 9) and a decentralized FL-based approach (cf. Figure 10) where each of the nodes represents an IoT client. The main idea of this approach is to establish a comparison between the traditional IDS approaches and the federated setting, where the centralized architecture will serve as baseline.

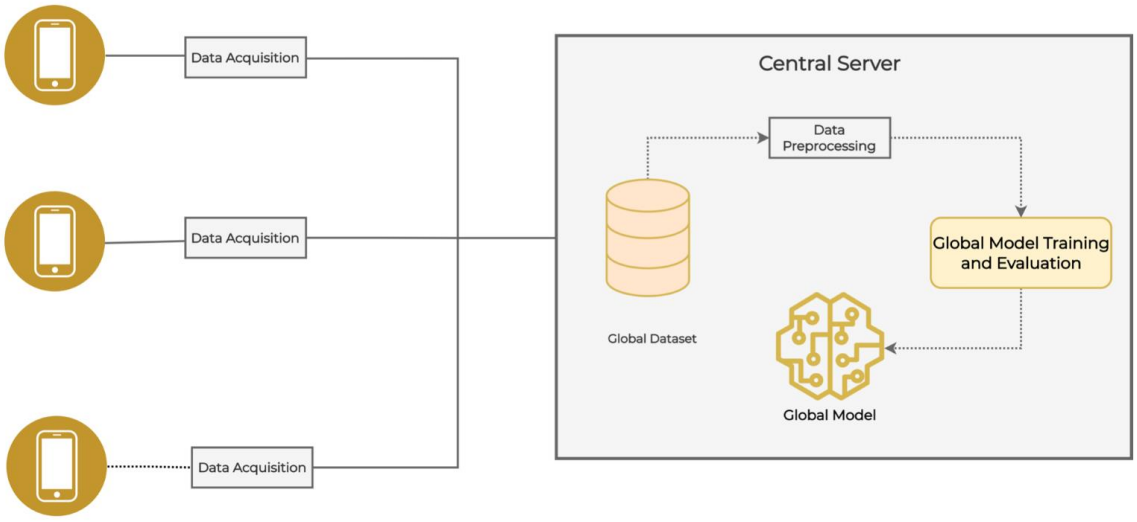

Figure 9. General Architecture of the centralized FL model.

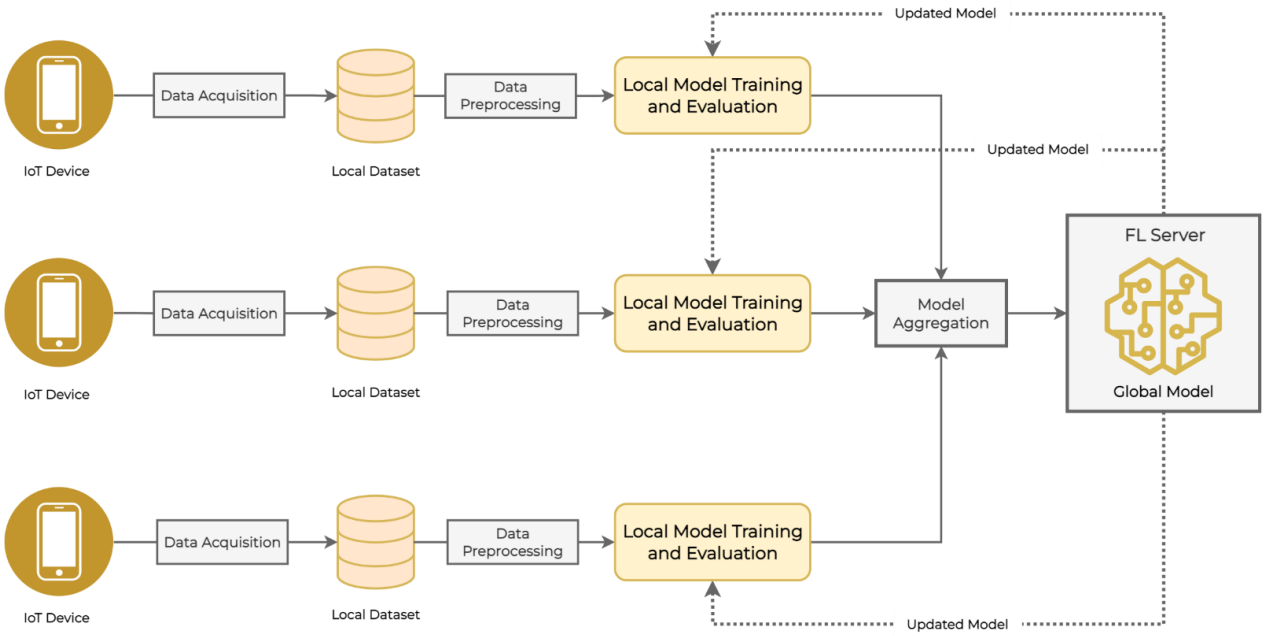

Figure 10. General Architecture of the decentralized FL model.

In the centralized setting, we will apply the following traditional ML algorithms that according to

the literature are the most successful: k-Nearest Neighbour, Random Forest, SVM, and Decision Trees.

For the Federated approach, we propose the FL algorithms that are going to be developed and evaluated in this scenario. *FedAvg* [11], as a baseline FL algorithm, *FedAvg* was chosen to implement, as it is simple and widely used in FL scenarios. The *FedProx* [12] algorithm was proposed as a way to handle the pending issue of heterogeneity, both statistical and between systems, in federated network scenarios, it could be a good candidate for the pending problem. This algorithm seeks to provide convergence guarantees when the data used in the learning step is non-identically distributed, referred to as statistical heterogeneity, and adhere to device-level constraints by allowing each client (device) in the network to do a variable amount of work, systems heterogeneity. *FedAvg* shows tight convergence rates, and it suffers from client-drift when the training data across devices is heterogeneous (non-IID), this can result in slow convergence rates. The *SCAFFOLD* [13] algorithm, (Stochastic Controlled Averaging for Federated Learning), has a similar objective to *FedProx*, it uses control variates (variance reduction) to correct the client-drift in its local updates. And finally, *FedDetect* [14], the main difference between *FedDetect* and *FedAvg* is the type of model that is used in the training phase. This algorithm was specifically proposed as a FL method of intrusion detection in IoT. The main idea of this algorithm is the use of a deep autoencoder as a way to detect anomalies. The deep autoencoder focuses on the reconstruction of the input data in an unsupervised learning manner. The autoencoder splits the neural network into two segments: the encoder and the decoder.

3.2.2.2 Evaluation approach

Our approach to evaluate the IoT HIDS will consider a contemporary dataset, ADFA-LD [15]. The data can be used to train ML models to detect whether there is an attack occurring on the system. This can be answered by classifying the traces as either normal or abnormal behaviour for a given system. Since there are only 324 possible system calls within a single trace, due to the operating system, we can view them as being the attributes of the raw dataset. Additionally, because the data is used to classify anomalies, we have two classes: anomaly or not.

There are many classification metrics for IDS (o assess the accuracy of the alerts). The first can be represented by a confusion matrix (Table 4) with the true positives, true negatives, false positives and false negatives, which can be used for evaluating the performance of an IDS. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class.

Table 4. Confusion Matrix.

	Normal	Intrusion
Normal	True Negative	False Positive
Intrusion	False Negative	True Positive

Additionally, the performance of the IDS will be measured by the following metrics such as the accuracy, True Positive Rate, False Positive Rate, False Negative Rate, and the Receiver Operating Characteristic (ROC). Furthermore, and in particular for the Federated scenario, we shall also evaluate the number of rounds that it takes to reach a certain level of accuracy.

3.2.3 Future work

The current work of this component involves experimental work and evaluation of ML models in a completely centralised setting. However, given that this component is closely tied to the Federated AI component, the next steps also involve the development and experimentation of a FL IDS prototype with some FL state-of-the-art algorithms, which we aim to achieve in the next months.

In addition to that we aim to integrate Behaviour Monitoring with the Federated AI component

from RISE, namely, the Data Rebalancer subcomponent, and also the Device Self Protection component. This will be followed by the implementation of the rest of the Behaviour Monitoring's subcomponents, like the Event Handler, for communication with other ARCADIAN-IoT components, and Data Extraction subcomponent, for direct extraction and parsing of host's logs in real time, Data Preparation subcomponent, for transformation of logs into model inputs, ML Model Unit subcomponent, for event classification, and implementation of the central model aggregator for the FL approach.

3.2.4 Current resources

Due to the early stage of development, there are no resources currently available to public. The source code of the Behaviour Monitoring component will be made available to the public once the development and evaluation are completed.

3.3 Cyber Threat Intelligence

The design and development of the Cyber Threat Intelligence component is part of **Task 3.4** (Cyber Threat Intelligence for IoT systems) in **WP3**.

3.3.1 Overview

3.3.1.1 Description

ARCADIAN-IoT intends to provide an instrument to gather, produce, elaborate, and share information regarding cyber threats and attacks in the IoT domain where end devices might be critically affected. The threat information is generally presented as Indicator of Compromise (IoC), and it can be shared between various partners to detect similar attacks in other organizations, or directly used to detect and analyse new security incidents.

RISE is building up an IoT-specific Cyber Threat Intelligence (CTI) system based on Malware Information Sharing Platform (MISP),[8] an open-source threat sharing platform, to orchestrate the process of (i) information parsing, formatting, and sharing, (ii) generation of IoCs, and (iii) fetching feeds to CTI. Although MISP already provides a bunch of useful functionalities to administrate cyber threat data, there are still some critical features missing, such as, IoC quality control, access control, and automatization which would be useful in IoT environments. Understanding the quality/reliability and timeliness of IoCs, having an appropriate level of contextualization in different organizations are challenging tasks. Thus, essential robust mechanisms are needed. Additionally, exploiting the Federated AI Component (Section 2.2), ML-based models will enable the sharing of IoCs among multiple instances in a privacy-preserving manner. CTI platform will enable the direct use of IoCs for anomaly detection, intrusion detection and prevention, and designing of novel protection mechanisms.

An IoT-specific CTI is responsible for collecting, processing, and sharing IoCs regarding to cyber threat within IoT network. Different features have been identified as needed for the essential functionalities to meet the use case requirements in ARCADIAN-IoT.

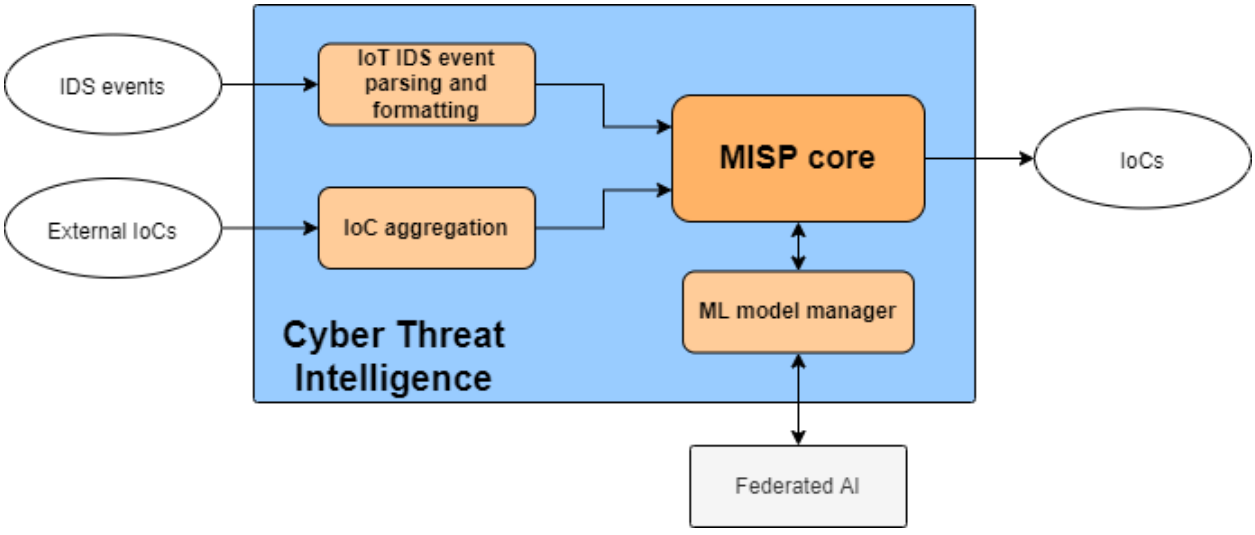

Figure 11. The architecture of CTI.

Figure 11 shows the architecture of CTI which consists of the MISP core, IoT IDS event parsing and formatting, IoC aggregation, and ML model manager. Each subcomponent is in charge of different tasks.

[8] https://www.misp-project.org/

- **MISP core**: It is the CTI engine that provides primary features of threat data gathering and sharing. It also includes extensive functionalities for automation and management.
- **IoT IDS event parsing and formatting**: It is the subcomponent that aggregates and analyses IDS events from devices in the IoT network to generate and share IoT-specific IoCs. Because traditional standards for cyber threat information (e.g., STIX, TAXII,[9] etc.) requires formats which are too heavy for resource-constrained IoT devices, we aim to develop a lightweight format and protocol that are suitable for communication of IoT-specific IoCs.
- **IoC aggregation**: It is responsible for the aggregation and pre-processing of existing IoT-specific vulnerability databases. It transfers the vulnerable information into IoC format and import them to MISP.
- **ML model manager**: It integrates the functionalities provided by the Federated AI component into the CTI which will enable privacy preservation while sharing threat information or trained models with third parties.

3.3.1.2 Requirements

A recall of the high-level requirements that have been previously defined and provided in deliverable D2.4 [2] is given below.

Requirement 3.3.1 – Threat data collection: The CTI should be able to collect threat data from various sources, local and internal sources (wide variety of various sources).

Requirement 3.3.2 – Private information: The CTI may need to share information about compromises; Local intelligence in application to not disclose sensitive/confidential information belonging to users or company.

Requirement 3.3.3 – Indicators of Compromise: The CTI should support Indicator of Compromise (IoC) generation and sharing by any participating IoT or edge device.

3.3.1.3 Objectives and KPIs

The aim of the ARCADIAN-IoT project is provide a MISP-based CTI platform focused on IoT-specific threat intelligence. With the aim of fulfilling the project's objectives, the CTI will be evaluated against the following KPIs:

- Support the common IoC sources (Opensource Intelligence, e.g., Shodan, social networks, and dark/deep networks).
- Support the common Intelligence formats (e.g., TAXII, STIX, CyBOX,[10] IODEF [16], OTX,[11] etc.).
- Enable that at least two stakeholders can receive shared data of threats/compromises.
- Promote sharing of IoT threat data in EU, respecting privacy and data regulations.
- Enable a novel automated and privacy-preserved CTI approach exploiting the European MISP platform (MISP4IoT).

All these KPIs are related to the main project objective "*Enable proactive information sharing for trustable Cyber Threat Intelligence and IoT Security Observatory*" (as recalled in Introduction).

[9] https://oasis-open.github.io/cti-documentation/
[10] https://cyboxproject.github.io/
[11] https://otx.alienvault.com/

3.3.2 Technology research

3.3.2.1 Technical findings and achievements

Despite technologies, tools, and best practices for threat data sharing are quite consolidated, automated processing of CTI platforms is an area where research is still advancing, especially in sectors including critical services, such as healthcare, banking, energy, and transportation. Along with the growth of IoT, several ongoing projects and standardization activities are working on new lightweight IoT security protocols, secure connectivity of IoT with cloud backend, distributed trust in IT, etc. However, CTI focused on IoT is a relatively immature discipline; in fact, most current CTI platforms focus on standard internet hosts.

In current reporting period, RISE has focused on launching MISP core and configuring it to meet requirements for different use cases. The main contributions in this task can be summarized as follows:

- Launched and configured MISP in RISE Cyber Range for the future exploit.
- Defined architectural details of the ARCADIAN-IoT CTI component and its subcomponents requirements.
- Implemented a set of extended functionalities for MISP core including automatic event/attribute fetching, updating, and searching.

In MISP, the structure of shared information is well-defined. The terminology in MISP can briefly be categorized into two classes: data layer and context layer. The former includes all the terms related to how the information is defined in MISP; the latter includes the terms referred to the relationship between different clusters of information. In details, the data layer contains:

- Event: The encapsulation for contextually linked information represented as attribute and object
- Attribute: The individual data points, which can be indicators or supporting data
- Object: The custom template for attributes
- Object reference: The relationship between other building blocks
- Sighting: The time specific occurrences of a given data-point detected.

And the context layer includes:

- Tags: The labels attacked to events/attributes from taxonomies
- Galaxy-clusters: The knowledge base item used to label events/attributes and come from Galaxy
- Cluster relationship: The relationship between Galaxy clusters.

The IoC, which is a piece of information that helps Intrusion Prevention/Detection Systems and their administrators to detect suspicious or malicious cyber activities, can be generated based on this data structure. IoC usually involves different attributes related to the object of an event, and it can be represented as network indicator (e.g., IP address), system indicator (e.g., string in memory), or bank account detail. However, there are no events, attributes, and objects yet specifically designed for IoT contexts. By analysing the three project domains and their specific requirements, we are able to define a uniform set of IoT-specific events, attributes, objects to include in IoT-specific IoCs.

The MISP platform includes three main roles in MISP in the management process: the general administrator, organization administrator, and publisher. CTI platform setup will include three different instances in three distinct locations controlled by RISE, IPN, and Truphone (TRU). In our current setup, RISE takes the role of general administrator, while IPN and TRU will be granted with organization administrator credentials in a second stage. The general administrator has the

highest authority to manage all the collected threat data and is in charge of monitoring how the information is being shared. Organization administrators have the authority to manage organizational threat events.

Through the MISP platform, a user can manually insert, edit, delete, and search for events or objects. Users also can define how to share the information by selecting one of multiple approaches provided by MISP, e.g., a user may define the profile of the organizations with which the information will be shared and the sharing time frame. The information included in the shared IoCs is constrained by the IoT-specific context and fulfils the requirement of threat data required by other components in the framework (MISP4IoT). MISP platform has been extended with automatic control functionalities (e.g., automatic insert, edit, delete, search). For example, we provide an API for automatic creation of threat events when an IDS event is raised by the flow/Behaviour Monitoring component.

The new functionalities have been implemented in Python by utilizing PyMISP library. PyMISP provides essential REST APIs to access to MISP platform. It allows us to develop automatic-control functions of MISP, and these functions are essential for future exploitation of the others subcomponents in CTI platform. Through these functions, the threat data can be automatically added, updated, deleted. The necessary information includes CTI platform's IP address, the content of attributes to the defined object, and the authorized key, and some additional parameters (e.g., threat level, distribution, IDS flag, etc.).

Besides the implementation, we have investigated the most suitable data formats and communication protocols for IoT domains. Starting from the well-known language and serialization format STIX (Structured Threat Information Expression), which is an open-source format and integrates well with MISP for exchanging CTI data, we aim to implement a simplified and lightweight version of it to exchange threat data within IoT domains. This format helps us easily contributing to and consuming from the CTI platform as all dimensions of suspicion, compromise, and attribution can be emphasized with descriptive relations along with the object identifiers. It is presented as JSON, which is a machine-readable format, and it can be visualized in a graphical representation. On the other hand, we exploit the TAXII (Trusted Automated Sharing of Intelligence Information) protocol, which was designed to support STIX, in order to tailor the methods used for sharing cyber threat information in the three ARCADIAN-IoT domains.

3.3.2.2 Evaluation approach

The current prototype is regarded as a preliminary step towards the integration with other components in the ARCADIAN-IoT framework. The evaluation metrics are aligned with the purpose of KPIs and aim at defining which and how many IoT-specific threats the CTI platform can handle, how many IoT protocols/technologies can be supported, and how many organization/stakeholders contributes to the information sharing via the platform. In addition, we will evaluate the component based on users' experience; after finalizing the prototype, we will conduct utilization surveys or questionnaires to obtain feedbacks from stakeholders. These feedbacks will help us in the cyclic research process of a continuous improvement of design and development.

3.3.3 Future work

Future work, which will be reflected into deliverable D3.2, will focus on the following: (i) Definition of custom IoC structure specifically for IoT cyber threats, (ii) Development of lightweight IoC format and protocol based on STIX and TAXII for IoT scenarios, and (iii) Development of the IoC aggregator for external resources. Integration and deployment of the Federated AI component and its subcomponents into the CTI will be performed in the last period of the project and reported in deliverable D3.3, along with the performance evaluation.

TRU will join the efforts of this component by bringing into the consortium the expert knowledge of their CSIRT (Computer Security Incident Response Team), who will have a node of the CTI

platform running on its premises to monitor and analyse any relevant threats.

3.3.4 Current resources

Due to the early stage of development, there are no resources currently available to public. The source code of the extended functionalities of the MISP core will be made available once all the evaluations are completed.

3.4 Network Self-protection

The design and development of the Network Self-protection component is part of **Task 3.5** (Self-healing and self-protection for IoT systems) in **WP3**.

3.4.1 Overview

3.4.1.1 Description

This component is responsible for providing the self-protection features required by the ARCADIAN-IoT framework. The component has to enforce a set of protection rules into the data plane with the aim of safeguarding the network infrastructure, the IoT devices, and services against volumetric DDoS. For this purpose, the component is hooked into the data plane providing a programmable enabler able to enforce security policies by means of protection rules. These capabilities are provided by the OpenVSwitch (OVS) Data Security Controller (DSC). The implementation is based on the well-known virtual switch OVS on which the UWS team is carrying out significant extensions and upgrades to support an enhanced programmability of the data plane to meet the expected requirements of the project. Thus, the component is being designed and implemented to cope with different data paths and network protocols related to IoT networks, overlay networks, or any other type of network involved in the ARCADIAN-IoT project.

This component, in cooperation with the Network Flow Monitoring (Subsection 3.1) and the Network Self-healing (Subsection 3.6) components, performs an autonomous cognitive loop able to detect and mitigate known cyber attacks (volumetric DDoS attacks). First, the Network Flow Monitoring detects the attack and raises an alert. This alert is received by the Network Self-healing that determines what action or set of actions execute to stop the attack (which rule(s) to enforce in the data plane and where in the network topology they will be inserted). Finally, the Network Self-protection is responsible for executing such self-healing rules in the data plane and thus, stop de attack and restore the network traffic back to pre-attack performance. The component architecture is composed of three main subcomponents. Figure 12 provides an overview of the Network Self-protection architecture. The functionality of the subcomponents is briefly described as follows:

- The self-management capabilities are achieved by the **OVS Protection Decider (PD)** which is in charge of deciding on every instant what subset of rules from the complete set of protection rules located into the PD will be enforced in the **OVS DSC** by creating an autonomous control loop to perform self-optimization of the protectional capabilities and thus achieving large scalability, really needed to deal with security on IoT networks. The PD is continuously monitoring the behaviour of the DSC in order to optimise its performance.
- The **OpenVswitch Datapath Security Controller (OVS DSC)** is the subcomponent that processes the network traffic and thus eventually enforces the protection rules into the data plane. Every packet through the data plane is deep inspected and classified, and an action is taken based on the subset of protection rules active at the moment. If a packet does not match any rule, it is sent to the PD subcomponent so the subset of protection rules kept in the DSC tables can be updated.
- Finally, the third subcomponent is the **Protection Control Agent (PCA).** It is responsible of providing a North Bound Interface (NBI) for the communication with the Network Self-healing component. The exposed NBI is an intend-based interface receiving technology-independent instructions that are translated into technology dependant commands to be enforced in the data plane. The PCA subcomponent allows a dynamic management of the life cycle of the set of protection rules in the self-management system: installation, modification, and deletion.

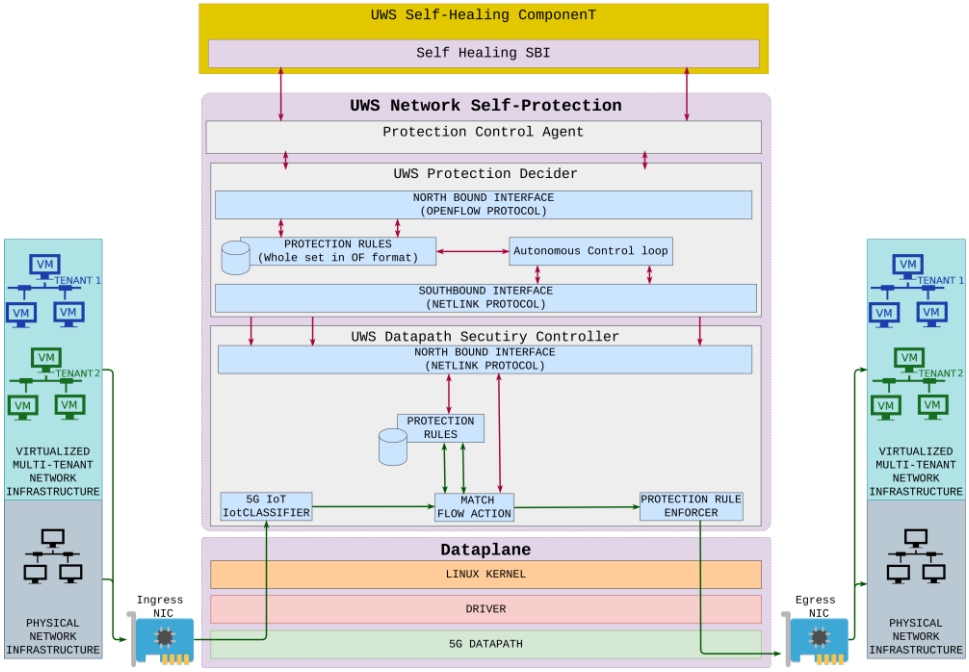

Figure 12. Network Self-protection architecture.

3.4.1.2 Requirements

A recall of the high-level requirement that has been previously defined and provided in deliverable D2.4 [2] is given below.

Requirement 3.4.1 – Mitigate Attack against IoT Overlay Networks: The Network Self-protection component will be deployed in the Edge and Core segments of the ARCADIAN-IoT infrastructure. In these segments, network traffic sent by IoT devices and sensors may be processed and encapsulated in overlay networks to guarantee user mobility and isolation between different tenants or users sharing the same physical infrastructure. Therefore, this component must be able to deal with overlay traffic and to identify the traffic from any IoT device regardless the number and types of encapsulation headers.

3.4.1.3 Objectives and KPIs

The aim of this component is to provide the Network Self-protection capabilities into the ARCADIAN-IoT framework. With the aim of fulfilling the project's objectives, the following KPIs will be used to assess and validate the performance of the Network Self-protection component:

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

All these KPIs are related to the main project objective "*Self and coordinated healing with reduced human intervention*" (as recalled in Introduction). Thus, this component will further contribute to mitigate and reduce the number of cyber incidents involving IoT devices.

3.4.2 Technology research

3.4.2.1 Technical findings and achievements

As a first step to the design and prototyping of the component, the following features have been identified as needed for the Network Self-protection component to meet the functionality and use cases requirements expected in the project:

- A Programmable Data Plane agent able to enforce dynamically security policies to mitigate known DDoS attacks in real time.
- Providing support for protection rules into IoT networks, overlay networks, and any other ARCADIAN-IoT related networks.
- Large scalability to deal with a vast number of IoT devices expected in 5G networks and the ARCADIAN-IoT use cases.

After a comparative analysis of the existing state-of-the-art software data paths, such as DPDK, XDP (eXpress Data Path) or OVS just to mention a few, OVS has been chosen as baseline for the implementation of the Network Self-protection component. The main reasons motivating this decision are the following:

- OVS is an open-source platform supported by a community workgroup keeping it in continuous development.
- It is a well-known software switch widely used in Software Defined Networks (SDNs) and virtualized networks where it is considered the de-facto standard.
- OVS provides a programmable OpenFlow NBI.
- It is widely used for frameworks such as OpenStack.
- Robustness and satisfactory performance.

The Network Self-protection component is being implemented on OVS version 2.16.2[12] which can run on Linux kernels from 3.16 to 5.8. We use the latest OpenFlow specification (version 1.5.1 released in 2015) [17].

The development of the Network Self-protection component involves a significant extension of the OVS base architecture and functionality aimed at fulfilling the requirements of the project. The following items have been identified during the design stage as extensions to be implemented in the OVS architecture:

1. Novel traffic classifier extended from traditional traffic classification for IP networks to a more complex classifier able to deal with all data paths expected in 5G IoT infrastructures such as overlay traffic with several levels of nested encapsulation.
2. Increase the expressiveness of the OpenFlow tables managed by the PD subcomponent with new fields providing a flexible, fine-grained flow definition fitting with the new fields extracted by the novel 5G IoT traffic classifier.
3. OpenFlow protocol extension aligned with 1) and 2) to provide a flexible fine-grained flow definition, enhanced control, and extended programmability to the Network Self-healing component.
4. Extension of the OVS OpenFlow NBI (ofproto library) functionality with the OpenFlow protocol extensions described in 3) allowing the PD to send and receive extended OpenFlow messages from the PCA.
5. Extension of the Netlink API for the inter-process communication between the Datapath Security Controller in kernel space and the PD in user space.
6. Upgrading of command line application suite included in the OVS distribution with the new capabilities to allow programmability via command line as an alternative method to Open Flow sockets.

The overall view of the Network Self-protection architecture designed by UWS is provided in Figure 12, which highlights the three main subcomponents integrating the functionalities

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

previously described: PCA, PD and Datapath Security Controller. Similarly, the internal and external interfaces are shown in the figure (a functional description of the interfaces is available in deliverable D2.5 [18]).

3.4.2.2 Evaluation approach

Following the prototyping, the overall behaviour of the Network Self-protection component will be empirically tested as a preliminary step towards the integration with the rest of the architectural components composing the Horizontal Planes of the ARCADIAN-IoT framework. The tests will be carried out in the cloud data centre at the UWS premises and will be focused on two aspects: functional validation and overall performance.

Regarding functionality, it will be empirically verified that the component is able to effectively enforce in the data plane the set of healing actions sent by the Network Self-healing component. The purpose of these actions is to stop malicious traffic coming from DDoS attacks previously detected by the Network Flow Monitoring component.

Regarding performance, the following features will be evaluated to demonstrate the suitability of the component for its integration into a 5G IoT architecture such as the one defined in ARCADIAN-IoT:

- **Scalability:** Maximum number of IoT devices traffic simultaneously managed.
- **Bandwidth:** Maximum data rate that the component is able to deal with.
- **Classifier overhead:** Comparative of the extra overhead introduced when processing different profiles of complex traffic (overlay networks with different number of nested headers).
- **Delay:** Average time to process each packet. That is, the time required to classify the packet and take a decision based on the current protection rule set enforced in the data plane. The default actions may be:
  - To drop traffic if it is identified as malicious traffic.
  - To process the packets according to the routing tables in the case of traffic labelled as legitimate.

  Any other kind of actions could be considered for the harmful traffic such as mirroring, redirecting to a Honeynet, sending to the SDN controller, etc.

Concerning the type of experiments executed, the component will be stressed against several challenging scenarios with different traffic profiles ranging several parameters:

- Packet size
- Number of IoT devices sending traffic
- Tx Bandwidth per IoT device
- Number and type of headers for tunnelling and overlay networks.

3.4.3 Future work

Currently the UWS team is engaged in the implementation of a functional prototype. A first version of this prototype will be available during the second year of the project.

After prototyping and as a last step in the development of the component, the prototype will be empirically validated and evaluated (as described in previous Subsection). Hence, a last version of the prototype will be provided in deliverable D3.2 and an empirical validation and evaluation of the overall performance of this component will be reported in deliverable D3.3. Finally, this component will be integrated in the ARCADIAN-IoT architecture, where it will interact with different architectural components. The major integration work will be undertaken with the Network Self-

healing and the Network Flow Monitoring components. These three components will cooperate to provide an autonomous cognitive self-healing loop in the network: The Network Flow Monitoring detects the attack and raises an alert; the Network Self-healing decides what action to take to stop and mitigate the attack and the Network Self-protection enforces the action in the data plane.

3.4.4 Current resources

Due to the application of IPR safeguarding measures, UWS has not the 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 Network Self-protection component will be released for the ARCADIAN-IoT consortium.

3.5 IoT Device Self-protection

The design and development of the IoT Device Self-protection component is part of **Task 3.5** (Self-healing and self-protection for IoT systems) in **WP3**.

3.5.1 Overview

3.5.1.1 Description

The IoT Self-protection component is responsible for providing protection at the IoT device level. This component enforces policies and protection rules to protect devices against attacks, preventing intrusions and other malicious threats from compromising the devices. To achieve this goal, IPN is building a solution that acts and protects the IoT devices based on threat information received in by other ARCADIAN-IoT components, such as the Behaviour Monitoring and CTI. Notwithstanding, this component is also able to enable self-protection mechanisms via self-imposed policies (e.g., loss of communication with other components).

An alternative approach for IoT device protection mechanism, will be pursued via eSIM-based device protection actions.[13]

Figure 13 depicts the components' internal modules – Event Handler, Policy Enforcer, and Self-protection policies – and its external interfaces. There are two main inputs: one with IDS events, from the Behaviour Monitoring components, and another with Indicators of Compromise from the CTI. In situations where the self-protection component enables significant self-protection measures, the self-recovery component is dully notified so that the device recovery process can be initiated – this is the only output of the component.

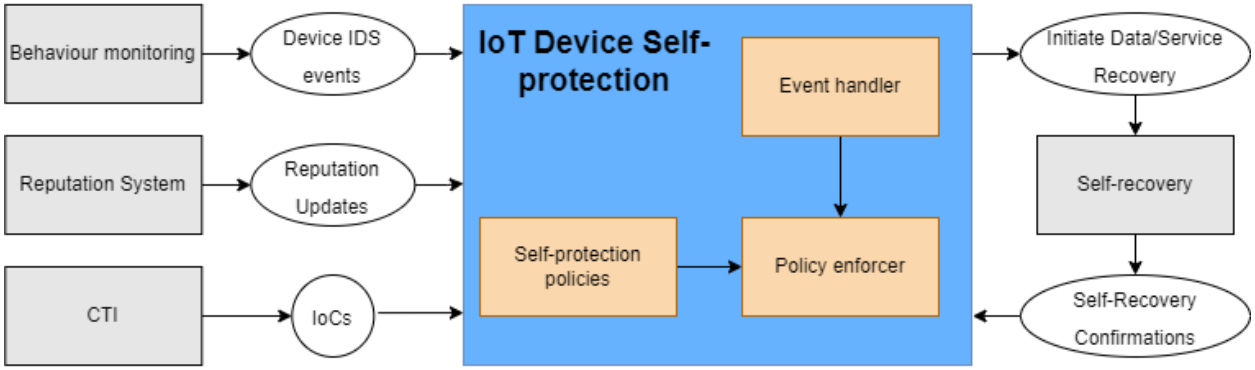

Figure 13. IoT Device Self-protection.

For each threat or malicious behaviour detected, the policy enforcer will act based on the policies locally stored (on the self-protection policies storage). The policy storage design enables policies updates (e.g., new policies or updated rules) whenever new threats are identified or for update and maintenance purposes. On the other hand, the event handler processes all incoming and outgoing events.

3.5.1.2 Requirements

A recall of the high-level requirements that have been previously defined and provided in deliverable D2.4 [2] is given below.

Requirement 3.5.1 – Heartbeat monitoring mechanisms towards ARCADIAN-IoT framework

[13] Details removed for potential future IPR protection measures.

should be implemented: The control over the traffic should provide the definition of protection/mitigation rules of IoT network infrastructures.

Requirement 3.5.2 – Device should allow at least one type of adaptive settings (e.g., dynamic configuration, rule enforcement, permission granting/revoking): The control over the traffic should provide the definition of protection/mitigation rules of IoT network infrastructures.

Requirement 3.5.3 – Device should provide administrative privileges to Self-protection component: The control over the traffic should provide the definition of protection/mitigation rules of IoT network infrastructures.

Requirement 3.5.4 – Device should be able to periodically obtain up-to-date classification of applications and services (e.g., from reputation system or CTI): The control over the traffic should provide the definition of protection/mitigation rules of IoT network infrastructures.

3.5.1.3 Objectives and KPIs

With the aim of fulfilling the project's objectives, the following KPIs were defined to assess and evaluate the performance of the IoT Device Self-Protection component:

- Provide at least two self-protection mechanisms (e.g., policy change, rule enforcement) for the supported devices.
- Support at least one self-protection mechanism (e.g., data encryption, policy change, rule enforcement) for scenarios with no connectivity.

All these KPIs are related to the main project objective "*Self and coordinated healing with reduced human intervention*" (as recalled in Introduction).

3.5.2 Technology research

3.5.2.1 Technical findings and achievements

The research work carried out within the scope of the IoT Device Self-protection component focused on analysing methodologies applied in related systems, in order to guarantee the protection of devices and data.

The state-of-the-art analysis has allowed us to define the most suitable approach for developing the proposed device self-protection solution. Specifically, the outcomes of this study, enabled the definition of: (i) an architecture representation suited for IoT Device Self-protection mechanisms; (ii) the target attack types and respective mitigation measures; (iii) a set of risk levels associated to different threats; and (iv) the proposition of eSIM actions as additional self-protection measures.

The following subsections describe each of the aforementioned aspects. The next section provides an overview of the reference architecture that the IoT Device Self-protection follows.

3.5.2.1.1 MAPE-K architecture

One of the known architectures that relates the most to self-protection mechanisms is the MAPE-K architecture [19,20]. The IoT Device Self-protection component will follow this approach and its subcomponents will directly represent MAPE-K architectural blocks. MAPE-K represents a cycle composed of five blocks: (i) monitor, (ii) analyse, (iii) plan, (iv) execute, and (v) knowledge. Each of the blocks has a specific role in threat detection. The roles can be described as follows:

- Monitor: Collects data from the software system as well as its execution environment.
- Analyse: Analyses the data gathered by the Monitor to see whether any variations related to self-protection have happened.
- Plan: Defines which system adjustments should be done to address the Analyzer's variances; Identifies the set of procedures or strategies that should be used in the system.
- Execute: Applies the changes identified by the Plan block in the system.

Knowledge: Provides information useful for self-protection, such as filtered data from the Monitor, statistical data, and self-protection strategies used in past incidents.

Figure 14 depicts MAPE-K cyclic architecture and its 5 blocks. The device is a central aspect of this architecture as it directly interacts with the first and last blocks of the architecture. The self-protection component is positioned in the blocks corresponding to the planning, execution, and knowledge. The planning block is built upon knowledge that is represented by **self-protection policies**. The execute block corresponds to the actions of the **policy enforcer** (i.e., a subcomponent of the IoT Device Self-protection). The monitor and the analyse blocks are assigned to the Behaviour Monitoring and CTI components, which are responsible for monitoring and analysing threats.

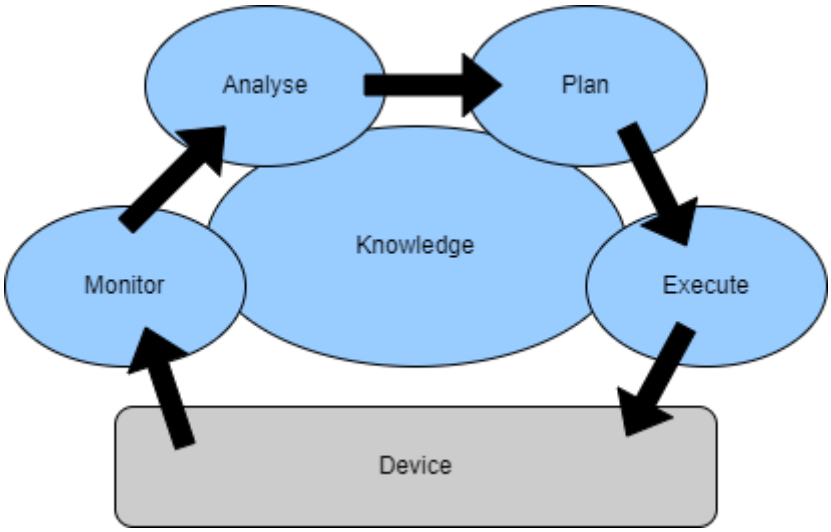

Figure 14. MAPE-K architecture flow representation.

3.5.2.1.2 Attack types and mitigation approaches

The IoT Device Self-protection component relies on Behaviour monitoring and CTI components to detect most device intrusions. The IoCs generated by the CTI platform will provide information associated to the event in question. As previously described, the Behaviour Monitoring will detect some attacks and the IoT Device Self-protection component will act in order to trigger self-protection mechanisms in order to mitigate or avoid the attacks. Some examples of attack types for which the IoT Device Self-protection component will provide mitigation measures are:

- Password Brute Force
- Add a new superuser (Client-side malicious executable)
- Code Injection (services running on devices)
- Authentication anomalies.

For such kind of attacks, it is possible to set up protection measures that mitigate or minimize the damage caused by such threats. The research carried out has allowed us to target, at least, the following mitigation measures:

- Close network ports to prevent intruders from entering
- Log out, suspend, or disable applications that are being threatened
- Use Two-Factor Authentication when the user's identity may be at stake
- Disable Root SSH Logins

- Use an allow list to limit access to specific Apps
- Notify users when suspicious device behaviour is detected.

In the coming months, it will be necessary to carry on with the work related to the mitigation of some threats. Furthermore, it will be necessary to further analyse the MISP taxonomy (described in Subsection 3.3), in order to implement a seamless and standardized way of communication between the CTI platform and the IoT Device Self-protection component.

3.5.2.1.3 Risk levels

Another aspect established during the research phase was the need to define appropriate security risk levels for various kinds of events, in the scope of cyber security risk management. In a proposal defined by Raibulet, C. et al. [21], it is possible to observe an example for the case of a home banking application (as shown in Figure 15). For several types of tasks, there are various levels of risk. As the risk increases, each of these risk levels presents an extra layer of security. In case a suspicious action (such as a high number of login attempts) is detected, the risk level for that action, and that particular account, is increased because it could mean the possibility of a brute force attack.

	Login			Bank Operations		
Risk level	Low	Medium	High	Low	Medium	High
Username	X	X	X			
Password	X	X	X			
OTP		X				X
Keybank			X		X	X
Secret Question			X	X	X	X
Captcha			X			X

Figure 15. Risk Levels – C. Raibulet et al.

For the IoT Device Self-protection component, this approach will be applied in a similar fashion. In case of threats where there is already prior knowledge of how to mitigate them, concrete and targeted actions will be defined (in self-protection policies) and applied according to the respective risk levels. Nevertheless, for threats or actions that are not yet documented or addressed (in the component's policy set), a differentiated parameter must be defined in order to indicate the threat level of such suspicious actions. In this way, it will be possible to devise a global response that aims to mitigate or minimize threats, regardless of the type of suspicious activity detected by the other components of the Arcadian IoT framework (e.g., device Behaviour Monitoring). The IoT Device Self-protection component's risk levels are still under formalization. This process considers the identification of the most relevant behaviours and actions on the devices which will then allow the formal definition of the risk levels and respective mitigation measures to apply at each of these levels.

3.5.2.1.4 Policy containers

To determine what kind of protection measures to apply to threats/intrusions reported to the IoT Device Self-protection component, it is necessary to structure specific actions in the form of self-protection policies. These policies should not only be lightweight but also adequately defined, interpretable, and accessible. It is possible to reach such goal by placing the self-protection policies in a database or in a file accessible to the component. In an IoT device - with limited computational resources - a structured file is the most efficient approach. Read and write file operations are fast and easy and do not require a database instantiation. These files will be structured in JSON, allowing structured loading of the protection policies, and enabling seamless updates of the protection policies.

3.5.2.1.5 eSIM role in device self-protection

Regarding the research on the participation of eSIM for device self-protection, the current results are the formulation of hypothesis for the use of the secure element, which has an ARCADIAN-IoT eSIM profile, and computational and storage capacity. Figure 16 depicts the specificity of the hypothetical vision formulated. The network authorization component is planned to receive the devices trustworthiness level and the related authorization policies from the reputation system component. When a device trustworthiness is reduced to a level that determines that it is compromised (policies and levels to be defined within the reputation system research), it distributes the trustworthiness information to the device secure element. ARCADIAN-IoT eSIM profile should be ready to receive this information and to perform automatic specific protection actions with it.[14] The communication happens securely, using over-the-air services accredited by GSMA-SAS.[15]

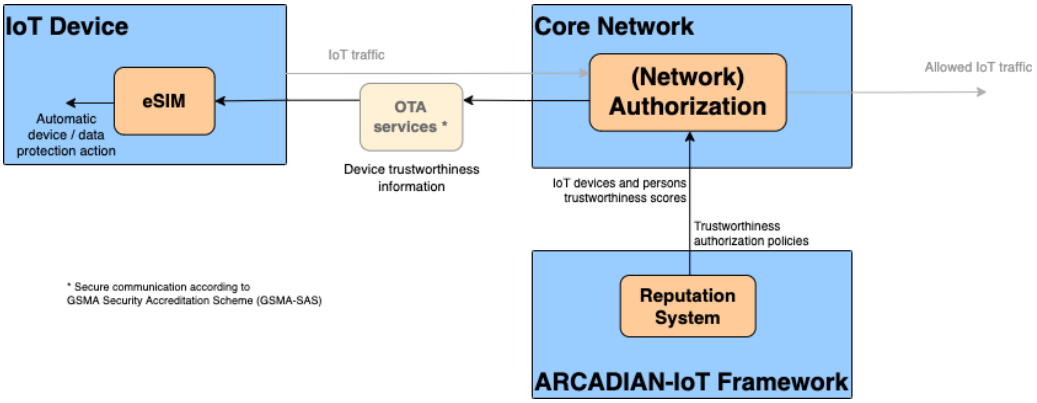

Figure 16. eSIM at device self-protection.

eSIM participation in IoT Device Self-protection adds a novel security ability that allows to have protection actions even when a device is non cooperative. For this we consider that the secure element is not only independent from the device itself (has its storage and processing unit with hardware isolation), but also has extremely secure communication channels with secure network servers that are able to inform it of the device trustworthiness. eSIM particular security actions were removed purposely for allowing future IPR protection but are planned to be delivered and demonstrated within the ARCADIAN-IoT context.

3.5.2.2 Evaluation approach

The evaluation of the IoT Device Self-protection component will consider various stages. For the first stage, it concerns communication and policy enforcing actions. As there is a great dependence on other components that will communicate intrusions and other suspicious behaviours (Behaviour Monitoring and CTI components) it is necessary to simulate the sending of intrusion detection events, or IoCs under at least the following circumstances: (i) normal device functionality; (ii) abnormal device functionality (i.e., not able to receive communications from the Behaviour Monitoring component running in the same device); (iii) no connectivity (no messages from the CTI platform and no interaction with eSIM network authorization service; and (iv) abnormal functionality and no connectivity.

It is necessary to test the policy enforcer subcomponent and the correct execution of protection policies under such scenarios and considering the different kinds of events and IoCs received. It is essential to simulate an actual attack and assess the efficacy of the self-protection actions applied by the component. The details are of such experimental setup are to be defined at a later

[14] Specific processes removed for allowing IPR protection measures.
[15] https://www.gsma.com/security/security-accreditation-scheme/

stage.

To further evaluate this component, it is also necessary to simulate the environment where this system will work. This will require the creation of an emulator that meets the hardware and software requirements for ARCADIAN-IoT compliant devices. This evaluation will be carried out on real devices under ARCADIAN-IoT domains (e.g., emergency and vigilance or medical IoT).

Finally, black-box tests will be performed, in which the intended behaviour will be compared to the expected outcome at the end of the execution. This must result from the response given by the Policy Enforcer to the threats found on the devices and must match the policy designated in the document.

3.5.3 Future work

After the research and approach definition carried out so far, the next step is to actively engage in experimental work towards the first functional prototype of the IoT Device Self-protection component. It will be necessary to configure the Event Handler subcomponent to start receiving messages from the Behaviour Monitoring and CTI components about intrusions and threats. In addition, a first version of the Policy Enforcer will be implemented, verifying which is the most appropriate policy from the self-protection policies document. This subcomponent will also be able to apply the corresponding response on the device to resolve the threats reported by Behaviour Monitoring and the CTI components.

We aim to provide the first functional prototype of this component in deliverable D3.2. It is anticipated to have limited functionality. Nevertheless, it should be able to apply a set of protection policies based on intrusions detected by the Behaviour Monitoring component. The last version of this component will be provided in deliverable D3.3. This version will applicate all the planned self-protection policies and the communication with the Self-Recovery component.

The eSIM technologies for IoT Device Self-protection are dependent on the network authorization component and the reputation system research. Therefore, future work will surely involve research collaboration with the related technical partners, namely IPN and UC. The eSIM technology for IoT Device Self-protection is expected to have the first functional prototypes delivered in D3.2. At a later stage, deliverable D3.3 will include the prototypes enhancement according to an evaluation done after D3.2.

3.5.4 Current resources

Due to the early stage of development, there are no resources currently available to public. The source code of the self-protection component will be made available to the public once all the evaluations are completed.

3.6 Network Self-healing

The design and development of the Network Self-healing component is part of **Task 3.5** (Self-healing and self-protection for IoT systems) in **WP3**.

3.6.1 Overview

3.6.1.1 Description

Within the scope of the ARCADIAN-IoT project, UWS is developing the Network Self-healing component. This component is designed to mitigate the potential impact of a cyber attack when protection rules for that kind of cyber attacks are not installed in the system (e.g., Firewall rules not installed) and thus the attack has the potential to penetrate the concerned IoT infrastructure.

To this end, the Network Self-healing component in ARCANDIAN-IoT will be based on an autonomous loop, where the following components and subcomponents are involved (see Figure 17):

- First, the **Network Flow Monitoring** component (Section 3.1) will provide the sensing and detection capabilities of a cyber attack such as DDoS. This will allow to detect the cyber attack.
- Second, the subcomponent ***Resource Inventory Agent (RIA)*** will perform the periodical reporting of the IoT network infrastructure with the intention to allow an effective self-healing decision-making process using the topological information.
- Third, the subcomponent ***Self-Healing Decision Manager (SHDM)*** will be in charge of determining what is the best plan to heal the network against that type of cyber attack, including advanced intelligence aspect related to the device on where to stop the attack, the interface inside of such device and the sense of the communication flow passing for such interface, to inform about what is the best effective way to perform the healing of the network and also to send such information to the associated self-protection component;
- Finally, the fourth component is the **Network Self-protection** component (Section 3.4) that will execute the mitigation of the attack, and finally deploying the necessary countermeasures to enforce the mitigation actions (e.g., traffic blocking/dropping, traffic mirroring, etc.).

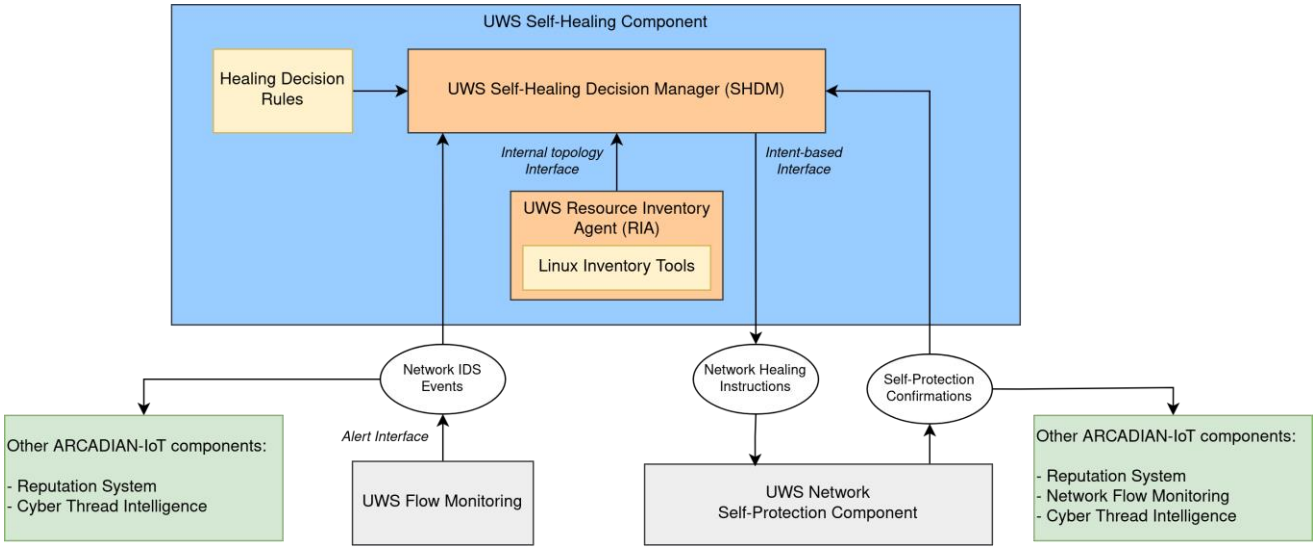

Figure 17. Architecture of the Network Self-healing component.

3.6.1.2 Requirements

A recall of the high-level requirement that has been previously defined and provided in deliverable D2.4 [2] is given below.

Requirement 3.6.1 – Distributed Healing: The Network Self-healing component will be able to perform protection/healing rules in a distributed way according to the topological information gathered from the infrastructure with the main intention to heal/protect the network against DDoS attacks.

3.6.1.3 Objectives and KPIs

With the aim of fulfilling the project's objectives, the Network Self-healing component will be evaluated against the following KPIs:

- Take the adequate decisions to stop an attack of up to 4096 IoT devices sending a combined bandwidth of 10 Gbps of malicious traffic in less than 20 seconds.
- Take the adequate decisions to stop an attack of up to 2048 IoT devices sending a combined bandwidth of 25 Gbps of malicious traffic in less than 20 seconds.
- Be suitable for deployment on the Edge and Core segments of the infrastructure as well as on dedicated management servers.
- Support at least 4 encapsulation and tunnelling protocols widely used in overlay networks inherent to 4G/5G IoT mobile infrastructures such as VXLAN, GRE, GENEVE or GTP, for instance.
- Recovery at least 95% of the system functionalities prior to anomalous behaviour.
- Support coordination of recovery to pre-defined trust levels.
- Reduce human intervention to the strictly required, in healing and recovery procedures.

All these KPIs are related to the main project objective "*Self and coordinated healing with reduced human intervention*" (as recalled in Introduction).

3.6.2 Technology research

3.6.2.1 Technical findings and achievements

The research carried out by UWS until this point has unveiled that the closest state of the art associated to the Network Self-healing component is the one related to Intrusion Prevention Systems (IPSs). These systems usually perform autonomous inspection of traffic and enforcing of rules to heal and protect the network if there is a cyber attack. Main problems with these tools are:

- They are traditionally designed for pure IP networks and thus they do not provide support for overlay networks nor for IoT network protection.
- They are usually installed in a component, which is deployed in the middle of the data path, and this is the only security control point available in the infrastructure and thus they do not provide support for dynamic distributed enforcing of protection/healing policies.
- They do not understand the network topology and thus are not able to take plans based on such topologies, especially the topology of IoT networks.

At this step we have designed the architecture of the Network Self-healing component based on the previously identified problems. The Network Self-healing component will provide Prescriptive Analytics using the data received in real time to determine which actions should be enforced in order to mitigate an alert. Thus, when an attack has been detected we need to know:

- WHAT action should be taken
- WHERE this action should be enforced
- WHEN it must be enforced
- For HOW LONG it must be active.

The designed architecture allows the Network Self-healing component (cf. Figure 17) to be integrated within a loop together with Network Flow Monitoring and Network Self-protection components. This loop provides the detection of alerts in overlay and IoT networks. These alerts will be detected in distributed instances of Network Flow Monitoring deployed along the IoT multi-tenant infrastructure. Thus, the Network Self-healing component will be in charge of receiving this information in a centralized instance. The information shared from the Network Flow Monitoring instances will include (i) information about the malicious flow, (ii) information about the rule that has raised the alert, and (iii) the point where it has been detected.

The Network Self-healing component is composed of two subcomponents: the UWS SHDM and the UWS RIA. RIA has been designed and a first prototype is being implemented. It is a distributed component that must be deployed at least in the same machines of the data plane where the Network Flow Monitoring and the Network Self-protection components instances are deployed. RIA is in charge of the discovery of the network topology, so that SHDM can identify the point where the alert has been raised and where it has to mitigate the attack. RIA uses the Linux Inventory Tools (cf. Figure 18) such as lldpcli, ispci, iproute2, lshw, brctl and ovs-vsctl to collect the topological information in the machine where it is installed. This information is reported periodically to an internal topology interface where SHDM is listening.

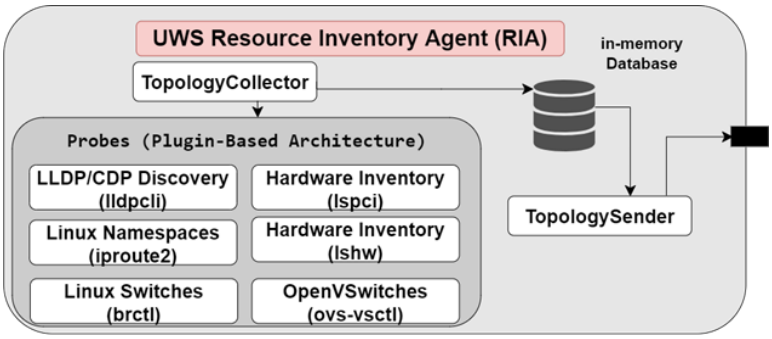

Figure 18. Architecture design of the UWS Resource Inventory Agent.

SHDM has been designed so that it will be a centralised component that implements the main functions of the Network Self-healing component. It will receive the alerts from the Flow Monitoring component and the topological information from RIA. SHDM will use a set of Healing Decision Rules to provide Prescriptive Analytics and autonomously decide the WHAT, WHERE, WHEN and HOW LONG parameters previously cited. This information will be sent as a Network Healing Instruction to be collected from the exact instance of the Self-protection component determined by the WHERE parameter.

3.6.2.2 Evaluation approach

Following the prototyping, the overall behaviour of the Network Self-healing component will be empirically tested as a preliminary step towards the integration with the rest of the architectural components composing the Horizontal Planes of the ARCADIAN-IoT framework. The tests will be carried out in the cloud data centre at the UWS premises in Paisley Campus (Scotland) and will be focused on two aspects: functional validation and overall performance.

Regarding functionality, it will be empirically verified that the Network Self-healing component is able to effectively receive the alerts raised from the different Network Flow Monitoring component instances, that the RIA subcomponent is able to discover the topology, that the SHDM subcomponent is able to produce Prescriptive Analytics and that the Network Self-healing enforce

in the data plane the set of healing actions sent by the Network Self-healing component able to communicate the Network Healing Instructions correctly with the final objective of stop malicious traffic coming from DDoS attacks previously detected by the Network Flow Monitoring component.

Regarding performance, the following features will be evaluated to demonstrate the suitability of the component for its integration into a 5G IoT architecture such as the one defined in ARCADIAN-IoT:

- **Scalability**: Maximum Number of devices in the network infrastructure to be discovered by RIA.
- **Delay**: Average time to process each received alert and produce the network healing instruction.

3.6.3 Future work

Currently the UWS team is engaged in the implementation of a functional prototype. A first version of this prototype will be available during the second year of the project.

After prototyping and as a last step in the development of the component, the prototype will be empirically validated and evaluated (as described in previous subsection). Hence, a last version of the prototype will be released in deliverable D3.2 and an empirical validation and evaluation of the overall performance of this component will be reported in deliverable D3.3. Finally, this component will be integrated into the ARCADIAN-IoT architecture where it will interact with other components to fulfil the project requirements. In this regard, it is particularly relevant the integration with the Network Flow Monitoring and the Network Self-protection components in order to achieve an autonomous cognitive self-healing loop in the network.

3.6.4 Current resources

Due to the application of IPR safeguarding measures, UWS has not the 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 Network Self-healing component will be released for the ARCADIAN-IoT consortium.

4 COMMON PLANE

The Common Plane provides the functionalities that are common to all other horizontal and Vertical Planes and includes the **Hardened Encryption** mechanisms aiming to provide a complete and innovative protection for IoT devices, and the **Permissioned Blockchain** which will provide immutable auditability and traceability properties to the data under management.

The research activity related to the components in the Common Plane is part of **Task 3.1** (Permissioned Blockchain) and **Task 3.2** (Hardened Encryption) in **WP3**.

4.1 Hardened Encryption

The design and development of the Hardened Encryption component is part of **Task 3.2** in **WP3**.

4.1.1 Overview

4.1.1.1 Description

ARCADIAN-IoT aims at providing encryption mechanisms to secure private IoT data. A standard approach is to employ symmetric cryptographic protocols allowing devices, servers, and users to secure their data for transmission, storage, or other purposes. Unfortunately, this approach implies that a big amount of cryptographic material, such as secure keys, needs to be managed. On one hand, keys could be leaked or breached, especially if a single authority server has access and stores all of them, while on the other hand, such a system does not offer much flexibility. ARCADIAN-IoT's component Hardened Encryption (HE) aims to overcome these limitations by providing a system that is more flexible, decentralized, and further hardened by one of the three supported options (depending on the scenario): (i) the hardware-based Root of Trust (RoT) provided by the eSIM component, (ii) the hardware-based RoT provided by an IoT device with an embedded crypto chip, or (iii) the hardware-based RoT provided by an independent / external crypto chip module integrated by vendor into its existing IoT device, as an add-on module.

XLAB is building the HE component consisting of three subcomponents. The first one is a software library that devices, servers, and other entities can use to encrypt or decrypt/access the data. The main paradigm we build on is so called Functional Encryption (FE), in particular a subfield of Attribute Based Encryption (ABE). ABE allows participants to secure their data based on policies that determine the right entity that can decrypt the data, on one hand; on the other hand, keys based upon the attributes of entities can be distributed. This introduces a flexibility into the system giving the choice on who can decrypt the data in the hands of the encryptors. Moreover, this drastically reduces the number of keys needed in the system.

Secondly, HE will include a decentralized key management system for distributing the (ABE) keys and access rights among the entities in the system. This will eliminate a single point of failure, such as a single server having access to all the cryptographic material. The key management system will be integrated with blockchain solution provided in ARCADIAN-IoT framework, and also with the framework’s authentication based on Self-Sovereign Identity and hardware-based RoT.

Finally, to increase the trust in the system and prevent impersonation attacks and similar breaches, the signing of encrypted payloads will be enabled by eSIM hardware-based RoT. Such signatures will guarantee the authenticity of data without the risk of cryptographic keys being exposed, due to being generated and embedded into the hardware secure element itself.

In scenarios where a crypto chip (as add-on module or embedded into an IoT device) is the adopted option, the IoT device’s firmware will be equipped with (i) an agent for handling the keys and (ii) an agent for handling the decentralized authorization (as a second protection, on top of hardware encryption); on the other side, the IoT middleware platform (specific for the selected scenario) needs to include features for (i) key management, (ii) ARCADIAN-IoT services management, and (iii) relaying on 3PP platforms and decentralised authorisation.

4.1.1.2 Requirements

A recall of the high-level requirements that have been previously defined and provided in deliverable D2.4 [2] is given below.

Requirement 4.1.1 – Encryption mechanism: Enable secure and lightweight encryption with access policy and hardware-based RoT.

Requirement 4.1.2 – Secure key-generation: Provide secure and scalable key management and delegation synchronized with the decentralized identity management.

4.1.1.3 Objectives and KPIs

With the aim of fulfilling the project's objectives, the Hardened Encryption component will be evaluated against the following KPIs:

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

All these KPIs are related to the main project objective "*Provide a Hardened Encryption with recovery ability*" (as recalled in Introduction).

4.1.2 Technology research

4.1.2.1 Technical findings and achievements

The research work carried out within the scope of the Hardened Encryption (HE) component focused on analysing the state of the art of the proposed technologies, researching the possibilities of intertwining them, and developing an architecture plan for the component and its interactions with other components.

4.1.2.1.1 Attribute-Based Encryption

Attribute-Based Encryption (ABE) represents a family of encryption schemes first introduced in 2005 by A. Sahai and B. Waters [22], that mathematically ingrains access control (specifically, access policies) directly into the encryption process itself. The family is commonly divided into two distinct subfamilies of schemes: Key-Policy ABE and Ciphertext-Policy ABE (CP-ABE) (see [23]). As the former is less suitable for use in this project, we focus on the latter. The principal idea of CP-ABE is that the ciphertext is encrypted with an access policy defining which attributes an entity needs to possess in order to decrypt the ciphertext. The access policy is normally in the form of a Boolean formula, i.e., the necessary attributes connected by the AND and OR logical connectives. The exemption of the NOT logical connective makes it impossible to exclude entities possessing more attributes than necessary, which is in line with the idea that entities may choose to hide parts of their identity in the Self-Sovereign Identity model. Entities wishing to decrypt the data encrypted with them in mind have to obtain the attribute/decryption keys (one per attribute) from the Attribute Authority that checks the entities' eligibility for attribute keys and delegates the keys to the respective entities. The data is then decrypted if and only if the set of attribute keys belonging to the entity satisfies the access policy. This procedure has the advantage that users need to maintain only their set of attribute keys to be able to decrypt all the data encrypted with them in mind, since data is always encrypted with the same public key (technically belonging to the Attribute Authority), instead of managing a complicated web of symmetric and asymmetric encryption keys.

A severe problem with ABE is the existence of a single Attribute Authority. The Attribute Authority needs to be a trusted entity, since it possesses the master decryption key (corresponding to the public key used for encryption) that can be used to decrypt all data encrypted with its public key,

and delegates decryption keys to all other entities in the network. It represents a single point of failure and is an obvious attack target for adversaries and malicious actors who wish to decrypt sensitive data, enrol their own devices into the network, or disturb the network by preventing the distribution of new decryption keys.

To decentralize the cryptosystem in ARCADIAN-IoT, we have developed an architecture plan to use CP-ABE encryption scheme that uses a multiple key authority setup (due to A. Lewko and B. Waters [23]) in order to mitigate a single point-of-failure weakness present in traditional ABE schemes. The scheme uses multiple Attribute Authorities that check the user's eligibility for decryption keys pertaining to presented attributes and then delegates the decryption keys to the user, as illustrated in Figure 19 below. The Attribute Authorities can serve decryption keys for a unique set of attributes, they can serve the decryption keys for the same set of attributes as other Attribute Authorities, or any combination of the two. The decryption policy can then be adjusted to require an almost arbitrary set of attributes from various Attribute Authorities for the user to possess in order to decrypt the data. The decryption policies can therefore be constructed in a way that it forces potential malicious actors to compromise many (or all) Attribute Authorities, not just a single one, in order to obtain a usable set of decryption keys, or in a way that maximizes redundancy to make the network resilient to DoS attacks. To the best of our knowledge, this is the first approach to use decentralized ABE schemes in the context of IoT devices.

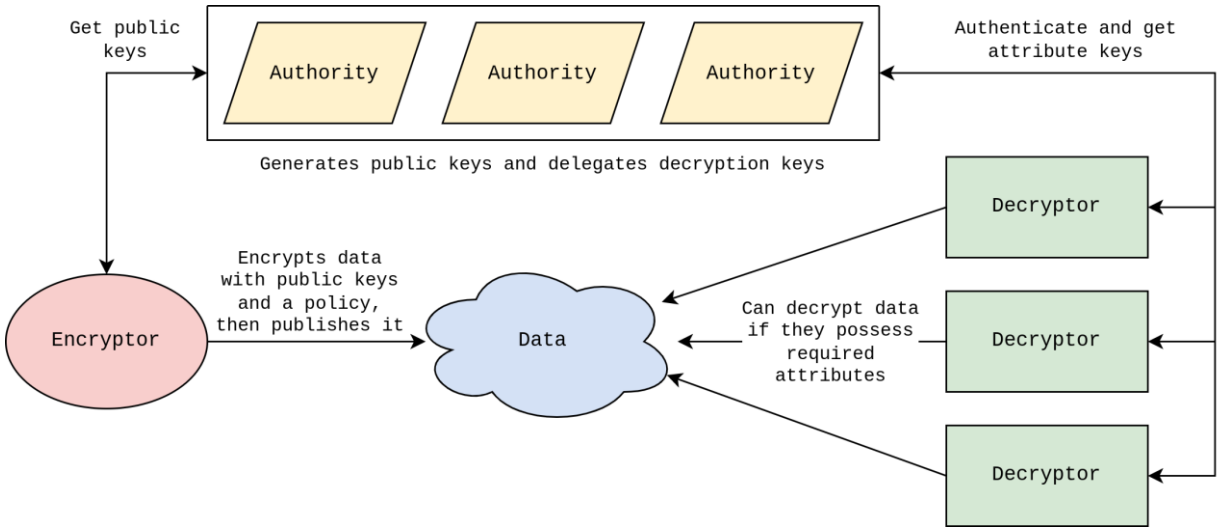

Figure 19. Attribute-Based Encryption with multiple authorities.

4.1.2.1.2 eSIM as hardware-based RoT for Hardened Encryption

Regarding the use of eSIM as a hardware-based RoT for the HE component, in the joint studies made by TRU and XLAB, several hypotheses were raised. Examples are: (i) having the encryption mechanisms within the secure element itself; or (ii) having it establishing a secure communication channel for IoT devices. Not considering ARCADIAN-IoT context, both hypotheses were feasible: the eSIM has cryptographic capabilities and allows the establishment of secure channels (e.g., (D)TLS) – see Figure 20 for details. However, considering the envisioned IoT solutions of the targeted domains, the first hypothesis was excluded due to the amount of data to encrypt (e.g., sets of photos from drones), which is unfeasible to be processed at the secure element. The establishment of a secure communication channel is a feasible approach but does not relate completely with the intended HE process that is described as an *encryption process with RoT information* (grant agreement). Considering the context and the objectives, the hypothesis selected for being prototyped for assessment and further research, is a novel combination of the eSIM abilities with the ABE previously described. It consists of using eSIM RoT for signing data encrypted at the device with the ABE (or the hash of that data). In this scenario, the ABE encryption in strengthened with the RoT signature, avoiding thus, e.g., impersonation attacks (malicious agents sending data – fake, corrupted or with other intention – in the name of other devices).

		IoT SECURITY APPLET 1	IoT SECURITY APPLET 2
TLS Version		(D)TLS 1.2 and 1.3	(D)TLS 1.2 and 1.3
Cryptography	RSA	Yes* (2048 bit)	No
	ECC	NIST P256	No
	ECDHE	Yes	No
	ECDSA	Yes	No
	PSK	Yes* (512 bits)	Yes (512 bits)
SHA-256		Yes	Yes
HMAC		Yes	Yes
HKDF		Yes	Yes

*optional

Figure 20. eSIM security abilities - GSMA IoT SAFE specifications.[16]

To this end, a novel eSIM applet, ARCADIAN-IoT eSIM applet, has been designed and is being prototyped. This artifact, whose general architecture can be seen Figure 21 (including a IoT SAFE applet, explained after) starts by being a regular eSIM profile for connectivity enablement, including the regular TRU's MVNO (Mobile Virtual Network Operator) connectivity features for such, like the eUICC OS or SIM OS, the NAA (Network Access Applications), and a set of applets that are able to run specific processes in the secure element, some of which specific from TRU (e.g. its multi-IMSI patented technology for programmatic management of worldwide connectivity attachment).

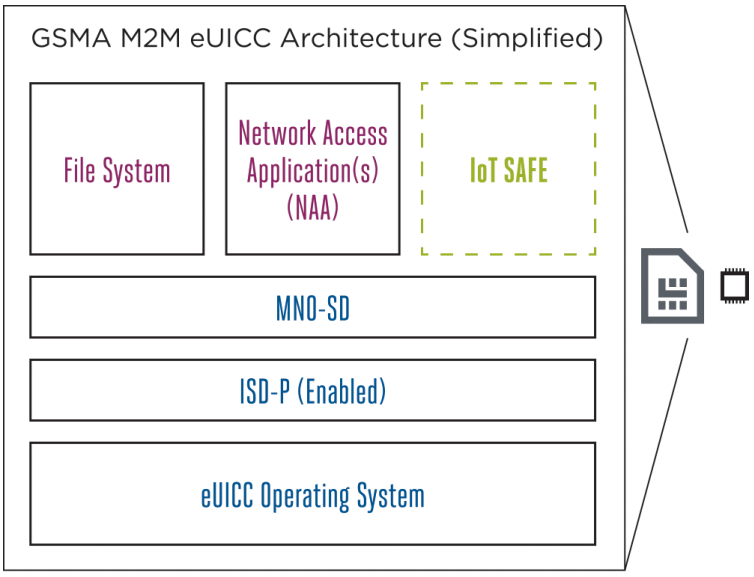

Figure 21. eUICC structure comprising the IoT SAFE applet.[16]

The novelty of ARCADIAN-IoT eSIM applet starts by its compliance with GSMA IoT SAFE[16] (SIM Applet For Secure End-to-End Communication) specifications integrated in a HE process, but it will be extended to novel attestation, device self-protection and device self-recovery processes,

[16] https://www.gsma.com/iot/iot-safe/

acting thus as RoT to ensure novel processes of device-to-cloud security and integrity. The research team will look forward to contributing to the enhancement of the existent state of the art, being already present in GSMA IoT SAFE working group to contribute with the outcomes of the research. Within the HE component, its first ARCADIAN-IoT eSIM prototype will have means to securely sign outgoing encrypted data with a secret generated and stored in the hardware secure element itself.

The intended functionality and the related processes for the first prototype are the following:

- *M2M bootstrap scenario*: When a compliant IoT device is turned on for the first time it is provisioned with an ARCADIAN-IoT eSIM profile (that includes the connectivity features and the security applet). Once the profile is activated in the device it connects to TRU network, which, recognizing the device and the ARCADIAN-IoT eSIM, requests, securely over the air, the security applet to generate a public/private key pair. The public key is returned, the certification process proceeds, and the public key is provisioned to the ARCADIAN-IoT components that will need to validate the signatures in the payloads.
- *Personal device bootstrap scenario*: When a user registers in an ARCADIAN-IoT compliant app it is provisioned with an ARCADIAN-IoT eSIM profile (that includes the connectivity features and the security applet). Once the profile is activated in the device it connects to TRU network, which, recognizing the device and the ARCADIAN-IoT eSIM, requests, securely over the air, the security applet to generate a public/private key pair. The public key is returned, the certification process proceeds, and the public key is provisioned to the ARCADIAN-IoT components that will need to validate the signatures in the payloads.
- *Hardened Encryption process*: To ensure data privacy and security, all the outgoing data from a compliant IoT device, or compliant app in a personal device, needs to be encrypted. The process consists of firstly encrypting the data (ABE), which then requests the RoT to sign the encrypted payload (or its hash), and finally sends it to the intended service.
- *Signature validation*: ARCADIAN-IoT services that are intended to verify the data integrity (e.g., attestation component, or the decryption component) should have the public key that allows to verify the RoT signature of the packages, ensuring thus its integrity.

The description above is a first high-level approach, to be enhanced in the next reporting period.

As previously mentioned, the previous scenario describes processes that relate with GSMA's IoT SAFE. This state-of-the-art specification,[17] which is still open for contributions, enables IoT device manufacturers and IoT SPs to leverage the SIM as a robust, scalable, and standardized hardware RoT to protect IoT data communications. Figure 22 depicts its components in the device side and in the server side.

[17] https://www.gsma.com/iot/iot-safe/

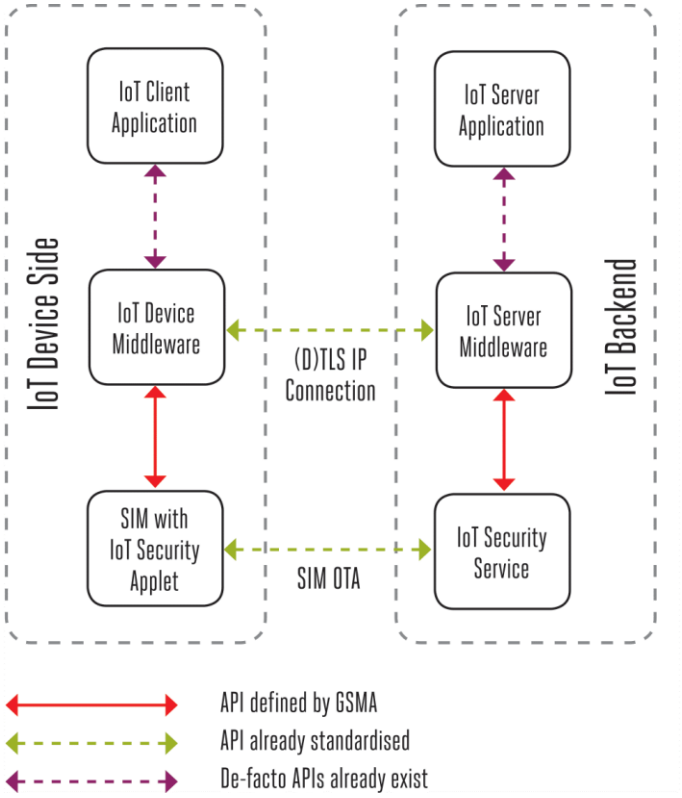

Figure 22. GSMA IoT SAFE components and their relation.[17]

In the first prototype of ARCADIAN-IoT eSIM profile, we are focusing on the security applet (the *SIM with IoT Security Applet* in the figure above). In the next reporting period, the IoT Device Middleware will be developed as well as the security services from the IoT Backend. The IoT Client Application is understood as being the HE process, and the IoT Server Middleware is the service that allows to securely communicate with the applet, interfacing with ARCADIAN-IoT framework.

Specifically, for the HE component, the first prototype of the RoT comprises the following methods, following GSMA IoT SAFE specifications.

Table 5. Selected methods for the first prototype of ARCADIAN-IoT security eSIM applet.

#	Method	Brief description [24]	Prototyping status
1	**Generate Key Pair**	The **Generate Key Pair** method is used to generate an asymmetric key pair. Upon successful execution both public and private keys are updated in the key store with their respective value and the public key is returned to the caller.	First prototype done in March-22. Potential current limitation: just allows one key pair per device (no key store implemented yet) Unit tests ongoing integrated with the compute signature – update method.
2	**Compute Signature - Init**	The **Compute Signature – Init** command opens a session to compute a signature. The command can also be used to cancel a signature computation session.	First prototype expected by April-22.
3	**Compute Signature - Update**	The **Compute Signature – Update** command is used to provide the applet with reference data to compute and return a signature to the caller	First prototype done in March-22. Unit tests ongoing integrated with the generate key pair method.

The methods in Table 5 are the key ones for fulfilling the intended functionality of the RoT in the

HE process. These will be target of unit testing and integration testing in the next reporting period, and the results will feed the research on the use of the RoT in the HE.

4.1.2.1.3 Crypto chip as hardware-based RoT for Hardened Encryption

When we aim to secure end-to-end grid sensors data traffic to/from a management IoT platform, the HE system consists of two parts: IoT device and middleware application. They are intended to provide protection to several types of industrial telemetry data in ARCADIAN-IoT, by employing modern and disruptive encryption mechanisms further hardened by a hardware-based RoT, the crypto chip in this case. The crypto chip has been designed in two different configurations, i.e., as add-on module or directly embedded into an IoT device. Also, the system will include a hardware encryption and decryption service for both authorisation, traffic, and recovery stages of IoT device lifecycle, with predefined configurable policies for interfacing with other related ARCADIAN-IoT services (as reputation, recovery, cyber security, flow monitoring, etc.).

The IoT device (containing the crypto chip) firmware provides (i) an agent for handling the keys and (ii) an agent for handling the decentralized authorization. The agent handling crypto keys is being developed from scratch, and embedded into IoT device firmware, following the data formats and specifications required by crypto chip supplying vendor. It will extract and validate the necessary keys for the appropriate steps in the authorization process and/or in telemetry (grid sensors) traffic management, and provide the primitives for running the encryption and decryption functions in the main firmware core system. The agent handling decentralized authorization will integrate an open-source protocol that will be chosen for this purpose, and will be embedded into the IoT device's firmware, respecting protocol specifications.

To work properly, the ARCADIAN-IoT HE component adopted to secure end-to-end communication in grid sensors infrastructures, requires the adoption of a specific middleware application running the following services: (i) a key management system (described in Section 4.1.2.1.4), (ii) an agent for managing the cooperation with other services provided by the ARCADIAN-IoT framework, and (iii) a set of interfaces to manage the interaction with 3PP systems and decentralized authorisation mechanisms (if this decentralised authorisation mechanism is not an Arcadian service, optionally could be a 3PP providing such).

4.1.2.1.4 ABE Key Management System

As explained in Section 4.1.2.1.1, the ABE keys in the selected protocol are manged with decentralized authorities. The developed architecture plan predicts deploying multiple servers in charge of distributing secret ABE key to authorized devices and users, such that they are able to decrypt appropriate information with respect to their attributes. The key management system architecture includes integration with other ARCADIAN-IoT components:

- The ARCADIAN-IoT blockchain ledger will provide a decentralized anchor of trust for public ABE keys (see Figure 23), as well as trusted decentralized hosting of the said public ABE keys directly.
- The authentication (possibly multi-level) system developed in ARCADIAN-IoT is used to access the key management system and request the needed keys.
- The system will enable attestation component by providing appropriate keys for the attestation process.

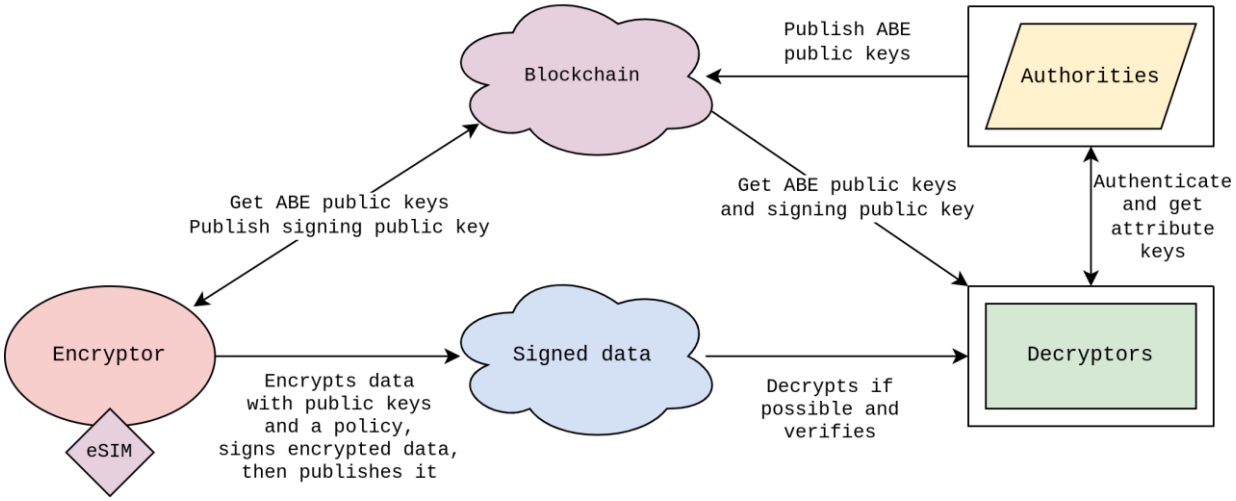

Figure 23. Attribute-Based Encryption architecture with integrated hardware RoT and blockchain support.

Furthermore, we will address other technical issues, such as techniques on transferring decryption keys over the network or over out-of-bound channels to entities other than the Attribute Authority that generates them (since decryption keys are mathematically points on predetermined elliptic curves and they need to be properly formatted and wrapped before transmission), and modelling the infrastructure needed to maintain a decentralized key management system composed of many Attribute Authorities using virtualization technology, including the Attribute Authorities themselves along with some peripheral and intermediate services, such as (TLS) certificate authorities and encryption proxies, respectively.

4.1.2.2 Evaluation approach

All the code associated with the HE component will be extensively tested with both unit and integration tests provided by the ecosystem of the programming language we are currently using (Go).

It is yet to be determined precisely how integration tests will be done with respect to integration with the other components of ARCADIAN-IoT, however high-level components such as blockchain network nodes and larger servers are likely to be simulated using virtualization technology. Test files and testing instructions will be provided together with the component and will be automated as much as possible using available testing frameworks and shell scripts.

In any case, the two parts of the HE component should be evaluated separately. The software library should be evaluated by running all the included tests and making sure the test coverage is as high as possible. The decentralized key management system should also be evaluated by checking its conformance to the relevant academic literature (which will be referred to in the component), and by observing its performance and security under heavy loads.

4.1.3 Future work

The scheme we are developing needs to be tailored to be deployed within the ARCADIAN-IoT framework, as its current state is generic and context-agnostic. IoT devices vary in computational power and capability, and this is reflected into the differences between devices in the Domains. The HE component will therefore have to be adjusted on a per-Domain basis as needed. For example, in Domain C devices will not perform their own encryption, but rather the data will be encrypted in users' phones, so the software library part of the component will need to be tailored accordingly. Other Domains use different cryptographic chips which will have to be taken into account.

In addition to an ABE scheme, a key-revocation scheme has to be implemented on top of ABE, since ABE schemes do not support revoking decryption keys natively. This is currently an active

area of study in the academic cryptographic community, since not allowing key revocation in public key cryptosystems with multiple entities poses a major security risk in case a malicious actor obtains valid decryption keys, or in case a legitimate entity turns malicious.

Working with other components, ABE keys must be integrated with the hardware RoT on the one hand and a blockchain ledger on the other (see Figure 23), taking into account the recovery components as well. ABE public keys especially should be included in the public identity of all Attribute Authorities on the blockchain so they can be retrieved in a decentralized manner. This will most likely be achieved by appending ABE public keys to the Attribute Authorities DID (Decentralized Identifier) documents, but depends on the Permissioned Blockchain component. The way the hardware RoT is achieved in a particular Domain will also influence how exactly ABE decryption keys will be handled.

The cryptographic infrastructure of Attribute Authorities has to be erected and integrated with other components. The ABE cryptographic infrastructure will most likely not be stand-alone but completely integrated into the blockchain network. The details are yet to be determined, as this will be a joint effort with other components.

A better system of writing ciphertext access policies for ABE should be implemented, since passing Boolean formulas is not suitable for use in a multiple authority setup, as it is unmaintainable and does not scale, in addition to being incomprehensible to the end user.

An additional avenue of interest to the HE component is the so-called Functional Encryption (FE), a generalization of ABE and several related cryptographic families. Instead of providing a mere access structure describing who can decrypt the data, but still decrypting it to its original form, FE allows the authority generating the decryption keys to generate specific function decryption keys that do not decrypt the data itself but rather compute the value of the specified function on the encrypted data and return the result in its decrypted form. The functions that can be chosen are limited to certain classes of functions, we namely have inner product schemes and quadratic polynomial schemes, which roughly correspond to computing a weighted mean of some data list and computing matrix multiplication, respectively. Furthermore, the data does not have to be encrypted by a single party but can be encrypted by multiple parties and then combined into a list before decryption. A particular use case that comes to mind is securing individual data-points of sensor data and only allowing the result of some statistical function to be decrypted by authorized entities. A significant difference between ABE and FE is that while the decryption access policy has to be known at the time of encryption of any piece of data (and cannot be changed after the ciphertext has been sent/published), the functions we would like to compute on the FE encrypted data do not need to be known beforehand. Any entity wishing to compute a different function on the FE encrypted data can simply ask the authority generating keys to generate them a new key for the desired function. In this way FE is much more flexible than ABE, but the authority has to be more careful when delegating keys not to give out sets of function keys that would allow one to compute the original plaintext data from function results.

4.1.4 Current resources

The GoFE functional encryption library written in Go, available publicly on GitHub,[18] will most likely act as a basis for the cryptographic components and the scheme we are developing. It contains implementations of many ABE and FE schemes. We have already extended the library with a basic implementation of the multiple authority ABE scheme mentioned above, which is planned to be used in the project. The GoFE library will be further expanded as needed and then either directly used or forked and modified to meet the needs of the HE component in particular Domains, in the development of the HE component's software library. In addition to containing the code, the GoFE library also refers to example uses of ABE and FE, which might come useful

[18] https://github.com/fentec-project/gofe

when developing the HE component's software library, and to a related C library which has similar functionality.

Multiple different projects that aim to introduce ABE to the world of IoT-devices offer a similar inspiration, e.g. [25]. A notable example of such a project is the solution brief on decentralized ID and access management for IoT networks published by the Hyperledger Telecom Special Interest Group and the LF Edge initiative at [26], and then successfully implemented by IBM and its partners. While most, if not all, of these projects use some form of blockchain technology, they often lack in providing a strong hardware-based RoT and in using novel cryptographic techniques such as ABE or FE to secure and verify data at rest and in transit. We aim to improve upon such solutions by bringing strong hardware security, modern blockchain technology, and safe implementations of new state-of-the-art algorithms together into a well-integrated component.

Most of the prototypes mentioned above are in publishing condition, but we are working on them with other components. As they mature, they will be released under an open-source license. The current prototype essentially consists of small pieces that each test one specific functionality of the envisioned component.

We already have a prototype of the ABE decryption key exchange over the network, it consists of two distinct programs, representing an Authority and a User (decryptor). The included test case involves setting up three instances of the Authority program and a single instance of the User program. The User program mutually authenticates with the Authority programs using PKI, then negotiates to obtain attribute decryption keys tied to their identity over the internet and stores them locally. The ABE parts of the exchange use the GoFE library. The described prototype will be made available on GitLab.

4.2 Permissioned Blockchain

The design and development of the Permissioned Blockchain component is part of **Task 3.1** in **WP3**.

4.2.1 Overview

4.2.1.1 Description

The Permissioned Blockchain is a common horizontal component in the ARCADIAN-IoT framework serving both Horizontal and Vertical Plane components such as Identity Management and Hardened Encryption, which make use of its immutable auditability and traceability properties.

Given the sensitive nature of data that will be shared in the network, ARCADIAN-IoT the project initially requested to use a private Permissioned Blockchain approach for all use cases, but this will again be re-evaluated upon deeper analysis of each use case. Permissioned Blockchains place restrictions on who is allowed to participate in the network and in what read or write transactions. It can have several conditional access features for users to obtain permission to operate at given levels.

In order to interact with the blockchain, software clients typically run their own node, automatically updating the common state internally and then notifying the rest of nodes. Importantly the Permissioned Blockchain has much higher throughput and aimed more at enterprise solutions whereas the public permissionless blockchains are much slower due to their global participation and complex consensus proofs.

4.2.1.2 Requirements

The high-level requirement that has been previously defined and provided in deliverable D2.4 [2] has been updated as shown below.

Requirement 4.2.1 – Provide a Permissioned Blockchain:

- To provide a Permissioned Blockchain to anchor the trust for Decentralized identifiers.
- To provide a Permissioned Blockchain to publish information to be shared in a trusted and immutable fashion with different actors in the ecosystem, e.g., reputation scores for things and persons.
- Users have the right to delete all personal information published on them stored on ARCADIAN-IoT components.
- To be able to support the deletion of personal and device information it is needed to make use of off-chain databases that have their trust anchored in the blockchain.

4.2.1.3 Objectives and KPIs

The primary objective is to deploy an open-source Permissioned Blockchain network to support the other components in the ARCADIAN-IoT architecture that will benefit from its decentralized immutable auditability and traceability properties. The blockchain component will be evaluated against the following KPIs in all use case domains:

- Have at least three components using the blockchain in the ARCADIAN-IoT framework.
- Facilitate deployment of blockchain technologies by non-cyber security experts in cyber security training sessions with, at least 20 participants.
- Deploy a blockchain network on at least 3 peer nodes.

All these KPIs are related to the main project objective "*Enable distributed security and trust in management of persons' identification*" (as recalled in Introduction).

4.2.2 Technology research

4.2.2.1 Technical findings and achievements

The primary characteristics that blockchain technology facilitates to components of the ACADIAN-IoT framework are as follows:

- ***Decentralized:*** Blockchain provides a decentralised peer-to-peer network of nodes that maintain an immutable record on a fault-tolerant ledger through consensus mechanisms. The decentralization means that all peer nodes have a copy of the ledger and access the same information so the greater number of nodes in the network the greater the fault tolerance. The consensus mechanisms facilitate that there is no central authority providing a trust anchor which facilitates the trust in the integrity of the data stored in the distributed ledger.
- **Transparent:** The decentralized network also means that any participant in the blockchain network can perform transactions with any other participant in a transparent manner. The transparency is achieved by the fact that all data on the ledger (on-chain) is available to all participants who have access to the ledger.
- **Private:** Privacy can be maintained by different means for example by providing a Permissioned Blockchain and also creating private blockchain subnetworks or channels where only those participants have access or by only storing hashes of private data on the ledger to be used to later check the integrity of private data stored (off-chain).
- **Immutable:** The intrinsic design of the recording blocks of data on the ledger provides for an incorruptible storage of data assuring its integrity from that point. This integrity of the blockchain also means that all transactions that enter data into the ledger are recorded and cannot be removed which has to be taken into consideration for storing of personal data due to the GDPR article on the “right to be forgotten.”

The ARCADIAN-IoT framework will be supported with a blockchain network of at least three peer nodes, as per the non-normative example below, with its component elements also described.

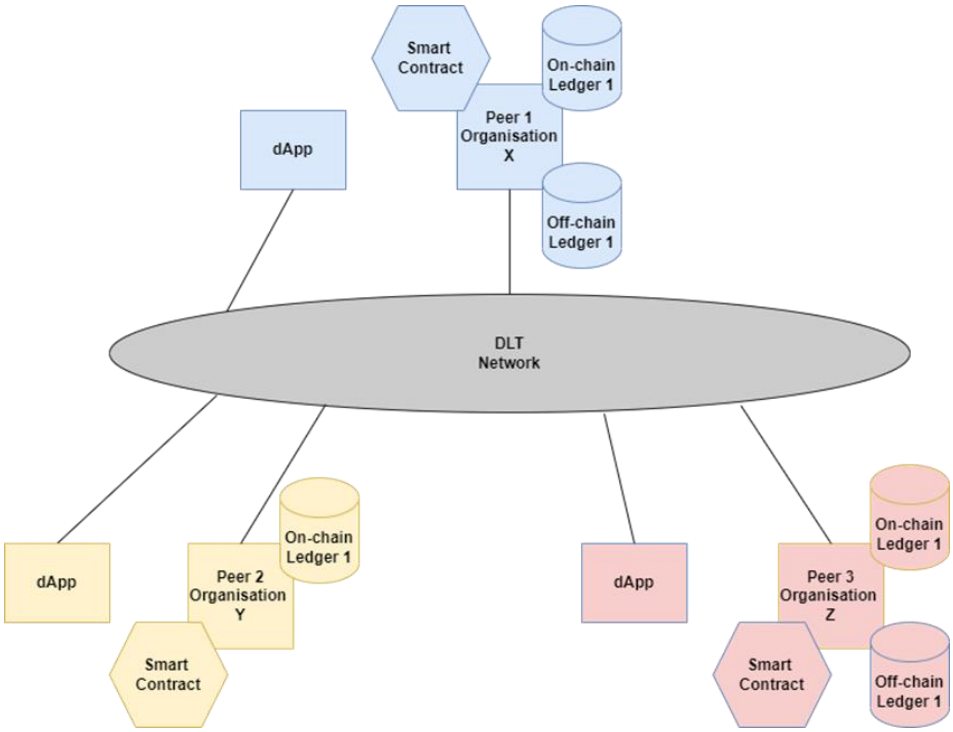

Figure 24. Example of a blockchain network with three peer nodes.

The blockchain network shown above is composed of three organisation deployments with the different colours representing each deployment. The functions offered by each element are described as follows:

- **Smart Contract:** This is essentially the backend business logic of a decentralized blockchain application running on the peer nodes. A smart contract functions as a trusted distributed application that gains its security and trust from the blockchain and the underlying consensus among the peers.
- **dApp:** This is the decentralized client application that requests the Smart Contract on the blockchain peers to carry out a transaction on a business object / asset and publish the result to the ledger.
- **Peer:** This implements the consensus protocol between the peers in the network for reaching consensus in publishing data to the ledger.
- **On-chain Ledger:** This is the immutable writing transaction on a ledger based on blocks of data protected by crypto proofs on previous block written on the ledger. Data written on the on-chain ledger cannot be deleted as deleting any block would violate the trust in the ledger.
- **Off-chain Ledger:** This is a data store where the integrity of the entries is provided by cryptographic hash stored on the blockchain. This is an important function for processing personal data and also data that needs to be shared only between a few private actors.

Note that the peer nodes will be hosted by trusted project partners who participate in the publishing of data on the blockchain.

The combination of blockchain characteristics, previously described, makes the blockchain useful for applications that would benefit from a decentralized trust model. Within ARCADIAN-IoT the primary use of the blockchain is to allow some trusted actors to write on the blockchain so as to publish data that is made available to its participants and shared with third parties which can then verify the integrity of the data against the blockchain. It is important to highlight that no personal data shall be stored on-chain.

The blockchains will be deployed in three main components of the ARCADIAN-IoT framework: (i) Decentralized Identifiers, (ii) Reputation System, and (iii) Hardened Encryption.

Since the data published on a blockchain is intrinsically dynamic and may also include personal information, it is normal practice that business objects/assets that need to be published should be stored off-chain while a crypto hash of the asset is instead stored on-chain.

Although a private Permissioned Blockchain might be preferred in some use cases, Decentralised Identifier methods for public entities (e.g., public persons, services, and things) are commonly deployed in publicly available blockchains. For this reason, ARCADIAN-IoT will deploy a Permissioned Blockchain network where only some participants will be able to write on it, but all members can read it; that is, both private and public Permissioned Blockchain networks with mechanisms of access control (stating which entities have restricted access) will be supported.

A high-level analysis of the state of the art of leading private Permissioned Blockchain technologies is carried out below.

Hyperledger Fabric[19]

Fabric is an open-source enterprise-grade permissioned Distributed Ledger Technology (DLT) platform established under the Linux Foundation and currently has reached a development community of over 35 organizations and nearly 200 developers. It is designed for the use in enterprise contexts, which delivers some key differentiating capabilities over other popular

[19] https://hyperledger-fabric.readthedocs.io/en/release-2.4/whatis.html

distributed ledger or blockchain platforms, and has a highly modular and configurable architecture, enabling innovation, versatility, and optimization for a broad range of industry use cases (including banking, finance, insurance, healthcare, human resources, supply chain and even digital music delivery).

It is the first distributed ledger platform to support smart contracts authored in general-purpose programming languages such as Java, Go and Node.js, rather than constrained domain-specific languages, so organisations benefit from existing competence.

The platform is also permissioned, meaning that, unlike with a public permissionless network, the participants are known to each other, rather than anonymous and therefore fully untrusted. This means that while the participants may not fully trust one another (they may, for example, be competitors in the same industry), a network can be operated under a governance model that is built off of what trust does exist between participants.

Key features of Hyperledger fabric are described in [27] and listed as follows:

- Rich queries over an immutable distributed ledger
- Support permissioned access to on-chain data on a need-to-know basis
- Support private data collections to protect personal and sensitive data off-chain
- Modular architecture supporting plug-in components
- Permissioned Membership Service
- Performance, scalability, and levels of trust
- Protection of digital keys and sensitive data.

Hyperledger Besu[20]

Besu is an open-source Ethereum client developed under the Apache 2.0 license and written in Java. It is designed to run on the Ethereum public network, but can also run on private permissioned networks, as well as test networks such as Rinkeby, Ropsten, and Görli. Hyperledger Besu supports several proof-of-authority algorithms (including QBFT, IBFT 2.0, and Clique) and proof-of-work (Ethash) consensus mechanisms.

Besu blockchain supports development of enterprise applications in Solidity to deliver secure, high-performance transaction processing in a private permissioned network.

Besu includes a command line interface and JSON-RPC API for running, maintaining, debugging, and monitoring nodes in an Ethereum network. You can use the API via RPC over HTTP or via WebSockets. The API supports typical Ethereum functionalities such as (i) Ether mining, (ii) Smart contract development, and (iii) Decentralized application (dApp) development.

Quorum[21]

Quorum provides a private permissioned network based on Ethereum for running smart contracts aimed at the banking and finance sector. Quorum strengths are high speed, scalability, and fast processing of private transactions. Its high speed is due to its simple consensus mechanisms (RAFT, IBFT Clique PoA, QBFT), and because of being an extension of the Ethereum platform; thus, most of the updates on Ethereum can be easily integrated in Quorum.

Quorum uses Solidity for smart contract developments; according to the DLT platforms comparison provided by Alastria,[22] Quorum is apparently less community active in recent years while there is heavy competition in the Enterprise Ethereum area and platforms such as

[20] https://besu.hyperledger.org/
[21] https://github.com/ConsenSys/quorum
[22] https://alastria-es.medium.com/comparison-of-dlt-platforms-be84950d339d

BlockApps and Hyperledger Besu.

Quorum networks can be used for a wide variety of use cases. In general, it seems to be the norm for banking and exchange-based applications, due to the benefits provided in terms of privacy brought to running processes, such as payments or post trade settlements.

IOTA Tangle[23]

IOTA Tangle is a DLT platform focused on IoT and provides a public permissionless network as well as, more recently, a private permissioned capability. IOTA nodes are thin, making possible to have an IOTA node running directly on a sensor. IOTA uses a persistence layer called IOTA Tangle which is a type of Directed Acyclic Graph with certain minor changes and a rival technology to blockchain. Whereas public blockchains rely on miners to select and aggregate the transactions with the highest fees into a sequential chain of blocks, the Tangle can process transactions in parallel, scaling with growing activity in the network. Consensus is also handled in small groups of nodes (and then rolled up to the rest of the network) which means that transaction times per second are extremely low.

Smart Contracts are supported in the ABRA language (supported by extensions like TOQN) and run on QBriq. The implementation of tokenisation smart contract complexity is relatively limited when compared to Smart contracts languages (e.g., Solidity and DAML). The network is effectively centralized as a central coordinator is used to manage the network; however, IOTA 2.0 (Coordicide) with its fully decentralized approach, will substitute its previous version soon.

4.2.2.2 Evaluation approach

The evaluation approach in deciding which DLT to employ in the ARCADIAN-IoT framework considered the following aspects:

- The support to different use cases (specifically ARCADIAN-IoT use cases and requirements)
- The running efficiency and optimal use of resources
- The ability to develop in commonly used development languages
- The open-source developer community
- The provided support documents and tutorials
- The off-the-shelf support of private data that needs to be kept off chain.

Upon analysis of the open source permissioned blockchains, and the identified use cases for its application, we have decided in the first instance to support ARCADIAN-IoT with Hyperledger Fabric. Hyperledger Fabric comes out very strong in all of the above mentioned aspects and in particular an efficient use of resources is achieved as most of the data can be stored off-chain, making use of its Private Data Collections while being notarized on-chain; also greater throughput and scalability are general characteristics of Permissioned Blockchains aimed at enterprise applications, due to only a few authorised organisations allowed to write to the blockchain, leading to more efficient consensus algorithms.

The methodology that can be used to evaluate Hyperledger Fabric smart contract transactions with benchmark figures is available on Hyperledger's GitHub.[24]

Ultimately, the evaluation of the Permissioned Blockchain in ARCADIAN-IoT framework and the smart contract(s) that are developed for its use will be obtained by the evaluation of the components that use it.

[23] https://www.iota.org/
[24] https://hyperledger.github.io/caliper-benchmarks/fabric/performance/

4.2.3 Future work

The next step will be to examine in more details the business objects (assets) that the ARCADIAN-IoT components has identified (cf. Section 4.2.2.1) for being published on the blockchain, their lifecycle, and who needs to publish and/or access them.

It will then be identified if there will be a general purpose, or application-specific, Smart Contract development for ARCADIAN-IoT and how this will be deployed in a blockchain network.

The technical design detailing implementation, deployment, and full external API specification for the integration in each of the domains (to be carried out in WP5) will be released in deliverable D3.2. The final technical design and software code will be provided in deliverable D3.3.

4.2.4 Current resources

No resources are available at the time of publishing.

5 CONCLUSIONS

The main objective of this deliverable is to introduce the set of ten components that belong to the Horizontal Planes of the ARCADIAN-IoT framework. In particular, it reports the preliminary findings from the research activities carried out during the first seven months of the project, including, but not limited to, the definition of the internal architecture in each component.

Each of the technical partners involved in WP3 (IPN, ATOS, MAR, RISE, BOX2M, UWS, XLAB, and TRU) has contributed with the preliminary stage of definition, design, and implementation of one or more components.

Each subsection is dedicated to a specific component which describes the component itself and the current status of the research related to it. As part of this work, partners have identified the methodology approach for evaluating the component, when considered as stand-alone.

Finally, we list the next steps in the development of the component together with the expected outcomes, at the end of every subsection.

The resulting technologies developed within WP3 will be integrated together with the components developed in WP4 for deployment and use cases demonstration in WP5.

REFERENCES

[1] D4.1: ARCADIAN-IoT Vertical Planes. ARCADIAN-IoT project (2022). Available from: https://www.arcadian-iot.eu/deliverables/
[2] D2.4: ARCADIAN-IoT framework requirements. ARCADIAN-IoT project (2021). Available from: https://www.arcadian-iot.eu/deliverables/
[3] De La Calleja, Jorge, and Olac Fuentes. "A Distance-Based Over-Sampling Method for Learning from Imbalanced Data Sets." In FLAIRS Conference, pp. 634-635. 2007.
[4] Mani, Inderjeet, and I. Zhang. "kNN approach to unbalanced data distributions: a case study involving information extraction." In Proceedings of workshop on learning from imbalanced datasets, vol. 126, pp. 1-7. ICML, 2003.
[5] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.
[6] Fernández, Alberto, Salvador Garcia, Francisco Herrera, and Nitesh V. Chawla. "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary." Journal of artificial intelligence research 61 (2018): 863-905.
[7] Wang, Han, Luis Muñoz-González, David Eklund, and Shahid Raza. "Non-IID data re-balancing at IoT edge with peer-to-peer federated learning for anomaly detection." In Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 153-163. 2021.
[8] Redana, Simone, Ömer Bulakci, Anastasios Zafeiropoulos, Anastasius Gavras, Anna Tzanakaki, Antonino Albanese, Apostolos Kousaridas et al. "5G PPP architecture working group: View on 5G architecture." (2019).
[9] Forrest, Stephanie, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. "A sense of self for unix processes." In Proceedings 1996 IEEE Symposium on Security and Privacy, pp. 120-128. IEEE, 1996.
[10] Haider, Waqas, Jiankun Hu, and Miao Xie. "Towards reliable data feature retrieval and decision engine in host-based anomaly detection systems." In 2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), pp. 513-517. IEEE, 2015.
[11] McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In Artificial intelligence and statistics, pp. 1273-1282. PMLR, 2017.
[12] Li, Tian, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. "Federated optimization in heterogeneous networks." Proceedings of Machine Learning and Systems 2 (2020): 429-450.
[13] Karimireddy, Sai Praneeth, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. "Scaffold: Stochastic controlled averaging for federated learning." In International Conference on Machine Learning, pp. 5132-5143. PMLR, 2020.
[14] Zhang, Tuo, Chaoyang He, Tianhao Ma, Lei Gao, Mark Ma, and Salman Avestimehr. "Federated learning for internet of things: a federated learning framework for On-device anomaly data detection." arXiv preprint arXiv:2106.07976 (2021).
[15] Creech, Gideon, and Jiankun Hu. "Generation of a new IDS test dataset: Time to retire the KDD collection." In 2013 IEEE Wireless Communications and Networking Conference (WCNC), pp. 4487-4492. IEEE, 2013.
[16] Danyliw, Roman, Jan Meijer, and Yuri Demchenko. "The incident object description exchange format." IETF Request For Comments 5070 (2007).
[17] Specification, OpenFlow Switch. "Version 1.5.1." Open Networking Foundation (2015). Available from: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf
[18] D2.5: ARCADIAN-IoT architecture. ARCADIAN-IoT project (2022). Available from: https://www.arcadian-iot.eu/deliverables/
[19] de AC Mello, Ruan, Admilson de RL Ribeiro, Fernando M. de Almeida, and Edward D.

Moreno. "An architecture for self-protection in internet of things." ICWMC 2016 (2016): 51.
[20] Glette, Kyrre, Peter R. Lewis, and Arjun Chandra. "Relationships to other concepts." In Self-aware Computing Systems, pp. 23-35. Springer, Cham, 2016.
[21] Raibulet, Claudia, Alberto Leporati, and Andrea Metelli. "Self-Protection Mechanisms for Web Applications." In Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering, pp. 181-188. 2016.
[22] Sahai, Amit, and Brent Waters. "Fuzzy identity-based encryption." In Annual international conference on the theory and applications of cryptographic techniques, pp. 457-473. Springer, Berlin, Heidelberg, 2005.
[23] Lewko, Allison, and Brent Waters. "Decentralizing attribute-based encryption." In Annual international conference on the theory and applications of cryptographic techniques, pp. 568-588. Springer, Berlin, Heidelberg, 2011.
[24] IoT Security Applet Interface Description. "Version 1.0." GSM Association (2019). Available from: https://www.gsma.com/iot/wp-content/uploads/2019/12/IoT.05-v1-IoT-Security-Applet-Interface-Description.pdf
[25] Pérez, Salvador, José L. Hernández-Ramos, Sara N. Matheu-García, Domenico Rotondi, Antonio F. Skarmeta, Leonardo Straniero, and Diego Pedone. "A lightweight and flexible encryption scheme to protect sensitive data in smart building scenarios." IEEE Access 6 (2018): 11738-11750.
[26] Solution Brief: Decentralized ID and Access Management (DIAM) for IoT Networks. Available from https://www.hyperledger.org/blog/2021/02/25/solution-brief-decentralized-id-and-access-management-diam-for-iot-networks
[27] Cocco, Sharon, and Gari Singh. “Top 6 technical advantages of Hyperledger Fabric for blockchain networks” (2018). Available from https://developer.ibm.com/articles/top-technical-advantages-of-hyperledger-fabric-for-blockchain-networks/