

APPROVAL

D5.5

Enhanced IoT Cybersecurity & Data Privacy/Trust

DRAFT

WORKPACKAGE WP5

DOCUMENT D5.5

REVISION V1.0

DELIVERY DATE 30/4/2023

PROGRAMME IDENTIFIER H2020-ICT-2020-1

GRANT AGREEMENT ID 957246

**START DATE OF THE
PROJECT** 01/10/2020

DURATION 3 YEARS

© Copyright by the IoT-NGIN Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 957246



DISCLAIMER

This document does not represent the opinion of the European Commission, and the European Commission is not responsible for any use that might be made of its content.

This document may contain material, which is the copyright of certain IoT-NGIN consortium parties, and may not be reproduced or copied without permission. All IoT-NGIN consortium parties have agreed to full publication of this document. The commercial use of any information contained in this document may require a licence from the proprietor of that information.

Neither the IoT-NGIN consortium as a whole, nor a certain party of the IoT-NGIN consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, and does not accept any liability for loss or damage suffered using this information.

ACKNOWLEDGEMENT

This document is a deliverable of the IoT-NGIN project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 957246.

The opinions expressed in this document reflect only the author's view and in no way reflect the European Commission's opinions. The European Commission is not responsible for any use that may be made of the information it contains.

PROJECT ACRONYM	IoT-NGIN
PROJECT TITLE	Next Generation IoT as part of Next Generation Internet
CALL ID	H2020-ICT-2020-1
CALL NAME	Information and Communication Technologies
TOPIC	ICT-56-2020 - Next Generation Internet of Things
TYPE OF ACTION	Research and Innovation Action
COORDINATOR	Capgemini Technology Services (CAP)
PRINCIPAL CONTRACTORS	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), INTRASOFT International S.A. (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Robert Bosch Espana Fabrica Aranjuez SA (BOSCHN), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), ENTERSOFT SA (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelxis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusätiö (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
WORKPACKAGE	WP5
DELIVERABLE TYPE	REPORT
DISSEMINATION LEVEL	PUBLIC
DELIVERABLE STATE	FINAL
CONTRACTUAL DATE OF DELIVERY	30/4/2023
ACTUAL DATE OF DELIVERY	28/4/2023
DOCUMENT TITLE	Enhanced IoT Cybersecurity & Data Privacy/Trust
AUTHOR(S)	Dmitrij Lagutin (AALTO) (ed.), Juuso Autiosalo (AALTO), Yki Kortensniemi (AALTO), Helmi Hirvelä (ABB), Henri Kinnunen (ABB), Dimitrios Skias (INTRA), Jonathan Klimt (RWTH), Y. Oikonomidis (SYN), Artemis Voulkidis (SYN)
REVIEWER(S)	Dimitrios Skias (INTRA), Terpsi Velivassaki (SYN)
ABSTRACT	SEE EXECUTIVE SUMMARY
HISTORY	SEE DOCUMENT HISTORY
KEYWORDS	cyber security, privacy, distributed ledger technology, interledger, self-sovereign identity, decentralised identifier, verifiable credential, digital twin, interoperability, ontology, semantic twin

Document History

Version	Date	Contributor(s)	Description
V0.1	6/2/2023	AALTO	Template for new document
V0.7	12/04/2023	AALTO, ABB, INTRA, RWTH, SYN	Partner contributions
V0.9	19/04/2023	AALTO	Ready for internal review
V0.95	27/04/2023	AALTO	Final version
V1.0	27/04/2023	CAP	Final quality check

DRAFT - PENDING EC APPROVAL

Table of Contents

Document History	4
Table of Contents	5
List of Figures.....	7
List of Tables.....	9
List of Acronyms and Abbreviations.....	10
Executive Summary	12
1 Introduction.....	13
1.1 Intended audience.....	14
1.2 Relations to other activities	14
1.3 Document overview	14
2 Overview of IoT cybersecurity and data privacy and trust in IoT-NGIN	15
2.1 Mitigation of poisoning attacks and early attack detection.....	15
2.2 IoT data privacy and trust	17
3 Early attack detection and mitigation	19
3.1 Malicious Attack Detector	19
3.1.1 Motivation for MAD	19
3.1.2 Description of the MAD solution	20
3.1.3 Results.....	23
3.2 Moving Target Defense Honeypots	28
3.2.1 Motivation for MTD Honeypots	28
3.2.2 Description of the MTD Honeypots solution	28
4 Semantic Twins.....	31
4.1. Motivation for Semantic Twins	31
4.1.1 Issues in IoT systems and Digital Twins	31
4.1.2 The role of Semantic Twins	32
4.2 Description of the Semantic Twin solution	34
4.2.3 Twin document.....	35
4.2.2 Discoverability and trustworthiness	36
4.2.4 Semantic descriptions.....	37
4.2.5 SSI API to Semantic Twin	38
4.2.6 Semantic Twin DLT integration	39
4.3 The twinschema.org Semantic Twin ontology.....	40

5 Decentralised Interledger solution	44
5.1 Motivation for Interledger.....	44
5.1.1 Need for multi-ledger transactions	44
5.1.2 Requirements of IoT-NGIN	45
5.2 Detailed description of the developed solution	47
5.2.1 Data flow of the Decentralised Interledger using High-Throughput Hyperledger Fabric DSM.....	49
5.2.2 Security properties of decentralised Interledger	54
5.3 Analysis of DIB features and performance	55
5.3.1 Initial DIB performance results.....	55
5.3.2 Ethereum vs. Hyperledger Fabric as the DSM layer	57
5.4 Practical use cases of DIB	57
5.4.1 Transferring hashes to more secure ledger.....	57
5.4.2 Using ledgers for Access Control.....	58
5.4.3 Trading of Virtual Assets	58
6 Self-Sovereign Identity Technologies	59
6.1 Verifiable Credential-based Access Control on Constrained IoT Devices	59
6.2 Triplet discovery using QR codes and GS1 Digital Links	60
7 Integrating the solutions	63
7.1 IoT devices configuration demo	63
7.1.1 Demo Description.....	64
8 Verification and Validation	67
9 Conclusions	68
10 References.....	69

DRAFT

List of Figures

Figure 2.1 – Attack detection and mitigation in FL systems (Solutions 1-4, shown in blue)...	16
Figure 2.2 – The IoT-device Triplet -related technologies developed in WP5	17
Figure 3.1 – GAN used for the Image generation of the malicious clients.....	20
Figure 3.2 – Overview of the poisoning attack and aggregation mechanism within the federated learning framework.....	21
Figure 3.3 – Accuracy of the classification model with different defence mechanisms.....	25
Figure 3.4 – Accuracy of the classification model with different defence mechanisms.....	26
Figure 3.5 – Confusion Matrix after applying the proposed mitigation technique.....	27
Figure 3.6 – IoT-NGIN MTD Honeypot framework overview.....	29
Figure 4.1 – A Semantic Twin describes a real-world entity and its Digital Twin forming an (entity) Triplet. The real-world entity may be a physical device, but also a more abstract real-world entity, such as an organisation.....	31
Figure 4.2 - A detailed look into an entity triplet, showing the basic features of a Semantic Twin and example composition of a Digital Twin. A Semantic Twin describes the whole System	33
Figure 4.3 – Example composition of a system of systems that uses different features of Semantic Twins.....	34
Figure 4.4 – Technical implementation of SSI API for Semantic Twins. Orange arrows depict the process of updating access requirements	38
Figure 4.5 – User interface showing the button to validate a document (a) and the result of the validation (b).	39
Figure 4.6 – Main Classes in the Semantic Twin Ontology.	41
Figure 4.7 – Application of the ontology to a fictional powertrain example use case.....	42
Figure 4.8 – The Ontology documentation browser on https://ontology.twinschema.org	42
Figure 5.1 – An IoT-based system combining multiple DLTs.	45
Figure 5.2 – DIB architecture consisting of nodes (Nx), bridge instances (Bx), and smart contracts (SCx).	48
Figure 5.3 – Interledger transfer process using Hyperledger Fabric DSM.....	51
Figure 5.4 – Recovery process of Interledger transfer.	53
Figure 6.1 – Overview of the SSI Access Control component.	59
Figure 6.2 – The triplet discovery protocol.....	61
Figure 7.1 – Illustration of the demo.	63
Figure 7.2 – Illustration of the DIDs used by the actors.	64

Figure 7.3 – The access control protocol to the Semantic Twin.....65

DRAFT - PENDING EC APPROVAL

List of Tables

Table 3.1 – Confusion matrix outcomes of multiclass classification.....	23
Table 3.2 – Parameters regarding the federated learning training process.....	24
Table 3.3 – Test set accuracy for the case of with/without mitigation.	27
Table 5.1 – Requirements for the interledger solution [D5.3].....	47
Table 5.2 – Initial performance results for DIB: Throughput and standard deviation (in parenthesis).....	56
Table 8.1 – Use of components in IoT-NGIN living labs.....	67

DRAFT - PENDING EC APPROVAL

List of Acronyms and Abbreviations

AAS	Asset Administration Shell
DIB	Decentralised Interledger Bridge
DID	Decentralised IDentifier
DLT	Distributed Ledger Technology
DNS	Domain Name System
DPoP	Demonstrating of Proof-of-Possession
DSM	Decentralised State Management
DT	Digital Twin
DTDl	Digital Twins Definition Language
FIB	Flexible Interledger Bridge
FL	Federated Learning
FOAF	Friend-of-a-friend
GDG	GAN-based Dataset Generator
GS1	Global Standards 1
IAA	Identity, Authentication, and Authorisation
IoT	Internet of Things
IVC	IoT Vulnerability Crawler
JWT	JSON Web Token
ML	Machine Learning
MLDT	Meta-Level Digital Twin [deprecated]
NGSI-LD	Next Generation Service Interfaces-Linked Data API
OData	Open Data Protocol
QR	Quick Response
SAREF	Smart Applications REference ontology
SOSA	Sensor, Observation, Sample, and Actuator
SSI	Self-Sovereign Identities
SSN	Semantic Sensor Network
ST	Semantic Twin
STA	SensorThings API
VC	Verifiable Credentials
WoT-TD	Web of Things Thing Description
W3C	World Wide Web Consortium

ZKP Zero-Knowledge Proof

DRAFT - PENDING EC APPROVAL

Executive Summary

This document focuses on some key problems of *cybersecurity, privacy preservation and trust improvement* in the domain of IoT systems, and presents the technical solutions developed in WP5 of the IoT-NGIN project to tackle these problems.

The work in WP5 has already been reported on in the earlier deliverables of the work package, so this deliverable now summarises and builds on top of the earlier documents. More specifically, in the deliverables D5.1 [D5.1] and D5.3 [D5.3], the requirements from the use cases in the IoT-NGIN project were identified and analysed to determine the best features and properties for the technical solutions to be developed in WP5. The State-of-the-Art technological solutions in the field of *local model poisoning attacks for on-device machine learning*, protecting the devices from *adversarial access*, *multi-ledger operations*, *semantic interoperability* practices for *Digital Twins*, and *Self-Sovereign Identities* were then analysed and, finally, the documents provided a high-level description of the solutions that were to be developed within WP5. Deliverables D5.2 [D5.2] and D5.4 [D5.4] then reported on the first versions of the developed solutions.

This document now presents a description of the final solutions. Specifically,

1. a GAN-based dataset generator for the creation of poisoned datasets that assist addressing attacks against IoT and Federated Learning systems
2. a Malicious Attack Detector (MAD) that facilitates the detection of cyberthreats and attacks against an IoT system
3. An IoT Vulnerability Crawler (IVC) that monitors IoT nodes and detects vulnerabilities
4. a Moving Target Defense (MTD) Honeypot Framework that deploys the honeypots dynamically
5. Semantic Twins that enable semantic descriptions of Digital Twins and the related real-world entities
6. a Decentralised Interledger Bridge (DIB) that enables transactions across different distributed ledgers (DLTs)
7. a privacy-preserving Verifiable Credential based decentralised on-device access control solution for constrained IoT Devices
8. a QR (Quick Response) code and GS1 (Global Standards 1) Digital Link based discovery mechanisms

The verification results of these solutions will be reported in the upcoming Deliverable D6.3 and the validation results from the IoT-NGIN Living Labs will be reported in Deliverable D7.4.

1 Introduction

The expanding use of IoT solutions has enabled many new services, but has also raised a range of new cybersecurity, privacy, and trust challenges. Ubiquitous IoT makes it possible to have a much more accurate and up-to-date situational awareness, but this can pose major privacy issues to the individuals, whose actions are being observed with this technology. Furthermore, individuals themselves are deploying more IoT devices and are in some cases even making the collected data available to a wider audience to enable new services, but at the same time also potentially raising privacy issues. Also, the increasing variety of IoT devices makes it harder to secure all the different types of devices against the many types of attackers ranging from individuals all the way to some governmental actors. Finally, for the audience utilising the data, a key question is which IoT devices and data to trust in this abundance of options.

Work Package 5 has developed 8 solutions to address the problems:

1. a GAN-based dataset generator for the creation of poisoned datasets that assist addressing attacks against IoT and Federated Learning systems
2. a Malicious Attack Detector (MAD) that facilitates the detection of cyberthreats and attacks against an IoT system
3. An IoT Vulnerability Crawler (IVC) that monitors IoT nodes and detects vulnerabilities
4. a Moving Target Defense (MTD) Honeypot Framework that deploys the honeypots dynamically
5. Semantic Twins that enable semantic descriptions of Digital Twins and the related real-world entities
6. a Decentralised Interledger Bridge (DIB) that enables transactions across different distributed ledgers (DLTs)
7. a privacy-preserving Verifiable Credential based decentralised on-device access control solution for constrained IoT Devices
8. a QR (Quick Response) code and GS1 (Global Standards 1) Digital Link based discovery mechanisms

For each of the solutions, the requirements and the State-of-the-Art of the technology were described in the first two reports D5.1 and D5.3 [D5.1, D5.3] and the first versions of the solutions were then detailed in the follow-up reports D5.2 and D5.4 [D5.2, D5.4].

This document provides a description of the final solution versions. For solutions 1-3, the development work was mostly completed and reported already in D5.2, so solutions 1-4 are discussed together in Section 3 to highlight how they interoperate. For solutions 5-8, there are more changes to report, so they receive multiple Sections (4-6) to first describe the individual solutions and then a Section (7) detailing how they can be used collaboratively.

The work will continue in WP6, which will report on the verification of the solutions in D6.3, and in WP7, which will report on the validation of the solution in the Living Labs in D7.4.

1.1 Intended audience

This document is intended for the following groups of people:

- Technical people interested in IoT systems, threat mitigation, decentralised applications, digital identity management, and Digital Twin interactions can find detailed solutions and some initial results in use cases.
- Solution designers and policymakers may find the document helpful to understand what kind of services the different technical solutions enable, which level of trust and privacy protection can be provided, and what standard ways for semantic interoperability are possible.
- Internal users within the IoT-NGIN project can find useful resources on the components or architecture solutions that are being made available in WP5, so that the use of developed components is made easier.

1.2 Relations to other activities

This document describes the technical solutions from WP5, and can, thus, provide guidelines to other work packages in the project on best practices in these fields. The following IoT-NGIN documents provide further information about the related project activities, which can be useful to extend the knowledge in this document. Architectural elements used in the IoT-NGIN project are described in Deliverable D1.2 [D1.2]. Deliverable D6.2 [D6.2] describes the architecture instantiations in each Living Lab (LL), as well as initial versions of the use case applications and initial testing and evaluation results. Deliverable D7.3 [D7.3] provides intermediate results about Living Labs use cases, including the components developed in WP5. The upcoming WP7 documents will also report on the use of WP5 solutions in the IoT-NGIN Open Calls and upcoming WP8 documents will cover the dissemination and exploitation of the solutions.

1.3 Document overview

The rest of the document is organised as follows.

Section 2 gives an overview on the discussed technologies and how they interact.

Section 3 describes Early Attack Detection and Mitigation solutions (Solutions 1-4).

Section 4 defines the concept of a Semantic Twin (Solution 5).

Section 5 covers the Decentralised Interledger Bridge (DIB) solution (6).

Section 6 presents two different Self Sovereign Identities solutions based on the use case requirements within the project (Solutions 7-8).

Section 7 describes how the solutions 5-8 mesh together to provide a comprehensive solution as depicted in the Installer app being developed in WP7

Section 8 briefly discusses how the solutions will be verified and validated in the upcoming deliverables.

Section 9 concludes the report.

2 Overview of IoT cybersecurity and data privacy and trust in IoT-NGIN

WP5 has developed altogether 8 solutions for addressing some key IoT cybersecurity, privacy, and trust problems. Of these, Solutions 1-4 focus mostly on cybersecurity while Solutions 5-8 focus more on privacy and trust. This section first presents how Solutions 1-4 complement each other and does then the same for Solutions 5-8.

2.1 Mitigation of poisoning attacks and early attack detection

In next-generation systems, AI solutions will be increasingly bound to IoT and edge nodes as AI solutions proliferate closer to the edge. While this enables more flexibility and reduced latency, the limited scope of knowledge any single node can generate over time means that the node's intelligence should be supported by sharing knowledge and experience with its peers. *Federated Learning (FL)* enables collaborative training amongst (federated) peers, allowing knowledge exchange while still protecting the privacy of the peers. It is realised by exchanging the parameters of ML models trained locally at each node, without disclosing the node's data to its peers. Although FL caters for data privacy and sovereignty, it can still suffer from malicious activity either due to compromised nodes or during model exchange. To address this, privacy preservation techniques in FL have been investigated within IoT-NGIN deliverables D3.1 [D3.1], D3.2 [D3.2] and D3.3 [D3.3], which aim to eliminate the impact of compromised input on the final machine learning (ML) model parameters.

But how can the attacks be detected or prevented in the first place? As part of WP5 work, the focus was on the detection and mitigation of attacks that may happen against IoT nodes participating in an FL system. The conceptual representation of the FL cybersecurity tools developed in IoT-NGIN is illustrated in Figure 2.1 (the IoT-NGIN solutions 1-4 appear in blue, while external entities are illustrated in grey colour). In this setup, the *FL system* (composed of a set of *Real devices*) is the one being protected from network or FL-specific attacks that try to compromise its operation. The Real devices have corresponding Digital Twin, which in IoT-NGIN is supported by WP4 IoT Device Indexing and IoT Device Access Control components (see IoT-NGIN deliverable D6.2 for details [D6.2]), and may also have Semantic Twins (covered later in this document). In this setting, WP5 FL cybersecurity tools take over the threat detection, monitoring, and mitigation processes for on-device FL systems.

As existing device vulnerabilities constitute opportunities for potential attackers, the *IoT Vulnerability Crawler (IVC)* (*Solution 3*) is responsible for scanning for potential vulnerabilities in networked devices, which may offer an entry point to "undesired visitors". The list of vulnerabilities is, of course, not static, but rather needs to be continuously updated. Therefore, IVC integrates OWASP ZAP [ZAP], which includes the classifications for the identified vulnerabilities of both the WASC Threat Classification [WASC], as well as MITRE's Common Weakness Enumeration [MITRE]. IVC also integrates the log4j_scan [LOG4J]. However, it should be mentioned that it follows a flexible plugin-based design which allows for easy integration of additional scanning tools. IVC has been described in detail in D5.2 [D5.2].

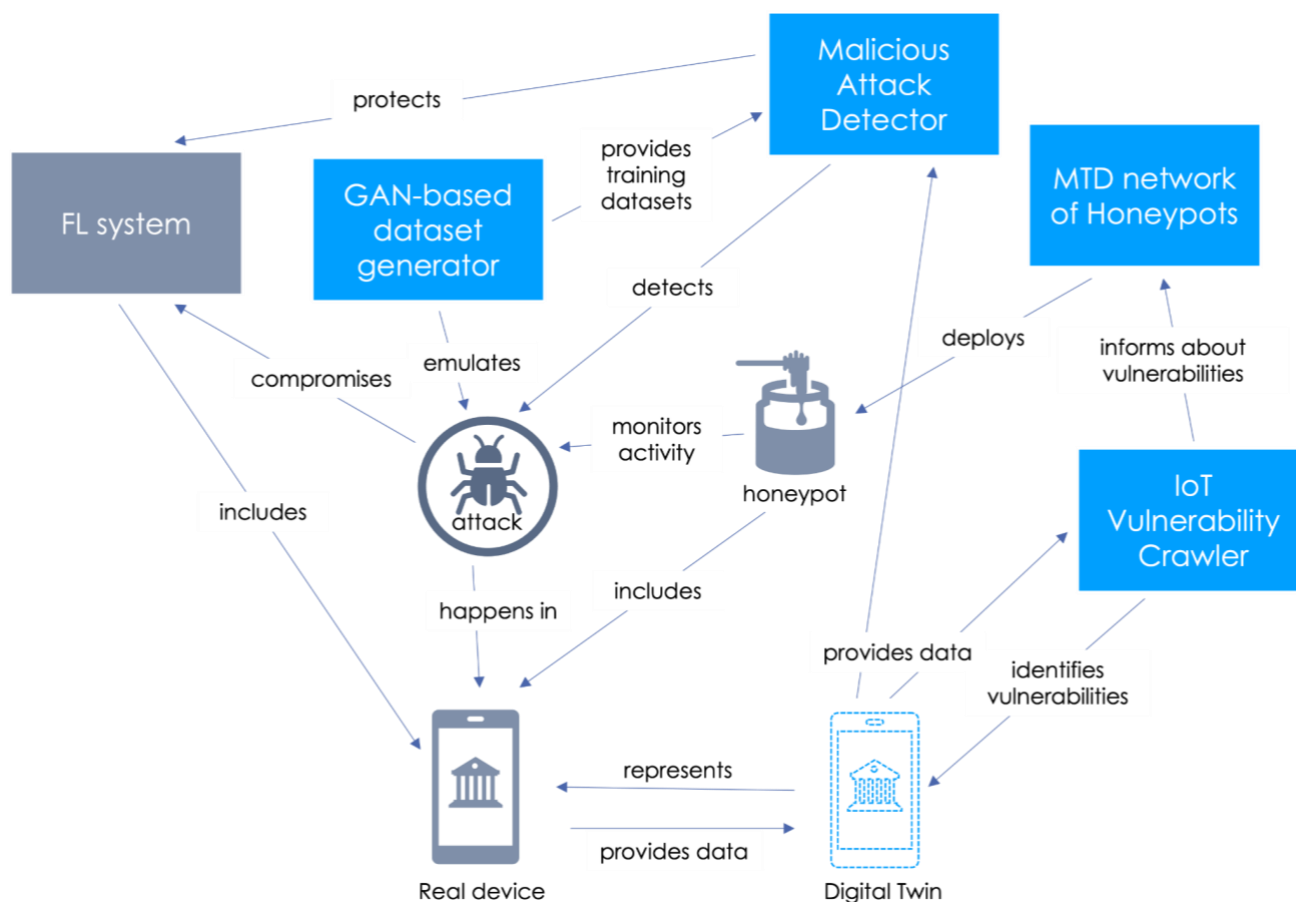


Figure 2.1 – Attack detection and mitigation in FL systems (Solutions 1-4, shown in blue).

Moreover, the identified vulnerabilities are further exploited in our approach in order to monitor the attack activity related to these vulnerabilities. Specifically, the *MTD Honeypots' Framework (Solution 4)* enables this by deploying a honeypot which leaves this vulnerability open for potential attackers. The monitored attack activity may provide new insights to the evolution of attacks and reveal new attack patterns, which would be relevant to the underlying IT/OT setup. The *MTD Honeypots' Framework* has been detailed and analysed in D5.2, while its role in early attack detection within IoT-NGIN is analysed in Section 3.

The *GAN-based Dataset Generator (GDG) (Solution 1)* assists in 'instructing' the FL system on how to detect and mitigate attacks against the FL nodes, both on the network level as well as on FL-specific attacks. At the network level, GDG is able to learn and generate datasets, which can then be used for training attack detection models. On the other hand, GDG is able to emulate sophisticated attacks in which the poisoned nodes act *maliciously from the beginning* while still imitating benign nodes as much as possible. This consistent behaviour makes the detection of the malicious behaviour harder compared to nodes that initially behave completely benignly and only after some time start behaving maliciously, as the change of behaviour is easier to detect. This functionality has been also exploited for the development of appropriate attack detection and mitigation processes. The full version of GDG has been detailed in D5.2.

The detection and mitigation of FL-specific attacks are implemented in the latest version of the *Malicious Attack Detector (MAD) (Solution 2)*. In its first version (detailed in D5.2), it was focused on the detection of network level attacks. In this deliverable, the detection of

sophisticated data and model poisoning attacks during FL training is now also possible. Further, MAD now incorporates mitigation for such attacks, which is based on identifying the poisoned nodes and excluding their contribution from the final model in FL training. This technique leads to robust FL training, even when 40% of the nodes act maliciously [Psy2023]. The new version of the MAD component is presented in Section 3.

2.2 IoT data privacy and trust

Much of the cybersecurity and privacy work in WP5 tasks T5.3-5 focuses on the IoT-device Triplet shown in the centre of Figure 2.2. The *Triplet* consists of a *real-world entity* (in this case, an IoT device), the *Digital Twin* (DT) that exposes the device's capabilities on the network, and the *Semantic Twin* (ST) that semantically describes the other two. When the real-world entity is something other than an IoT device (e.g. a shopping mall or a person), the Triplet can also be called an Entity Triplet, but in IoT-NGIN the focus is mostly on Triplets with IoT devices.

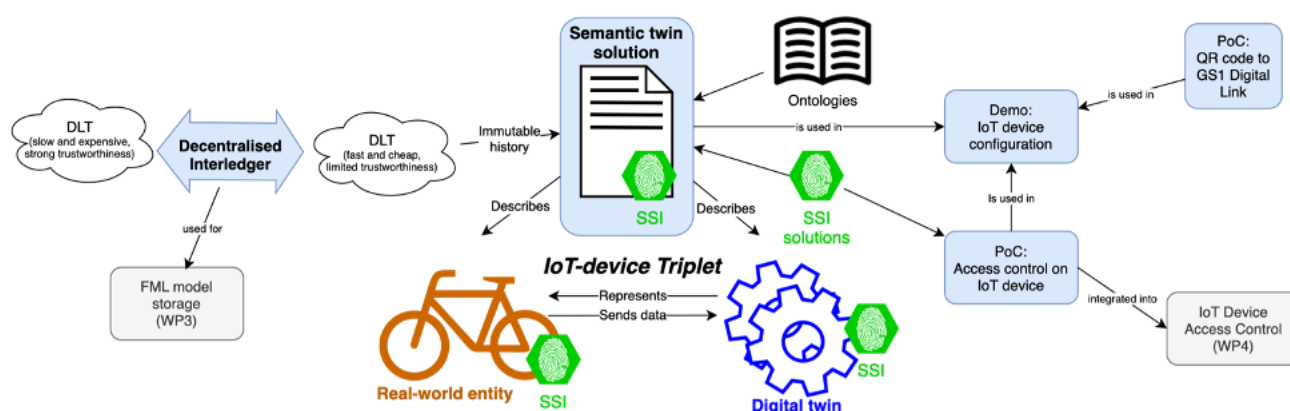


Figure 2.2 – The IoT-device Triplet-related technologies developed in WP5.

To support the IoT-device Triplet, WP5 has developed multiple solutions, as shown in blue in the figure. First, the *Semantic Twin (Solution 5)* is a novel concept of providing a structured semantic description of the Triplet. The core element is describing the capabilities of the IoT device and Digital Twin and where they can be accessed. This information can then be complemented with many other types of information, e.g. the licensing of the services and where access could be purchased, information about the validity of the services through, e.g. 3rd-party certification, etc. To make this semantic information as machine-readable and interoperable as possible, the information is organised based on ontologies, particularly *Smart Applications REference ontology* (SAREF) ontologies that are aimed for IoT use cases. The Semantic Twin is detailed in Section 4.

Another solution developed in WP5 is a *Decentralised Interledger Bridge (DIB) (Solution 6)* that allows us to link distributed ledgers (DLTs) and blockchains with atomic transactions. There are multiple interledger solutions, but most of them only focus on financial transactions or have limitations on the types of DLTs/blockchains they support as described more in detail in Deliverable D5.3 [D5.3]. IoT-NGIN has focused on a bridging-type interledger, which supports a broad range of ledgers and is agnostic of the transaction type, so it can be used with almost any type of application. Specifically, the work builds on an existing centralised bridging solution, which provides suitable functionality and interfaces, but suffers from the limitations of a centralised solution, namely higher trust requirement on the party running the

bridge and lower resiliency. IoT-NGIN has, therefore, developed a decentralised version of the technology, the *Decentralised Interledger Bridge (DIB)* described in Section 5, which allows us to overcome these limitations by utilising the same decentralisation approach as the DLTs and blockchains themselves rely on. With the interledger, e.g. the Semantic Twins can now rely on multiple ledgers to provide a shared immutable history in a cost-effective manner.

To improve the privacy of the people utilising the Triplet, our work leveraged Decentralised Identifiers (DIDs), an identifier technology that follows the Self-Sovereign Identity (SSI) principles. An SSI identity owner should be able to generate and use as many anonymous identifiers as they need to protect their privacy, e.g. to prevent correlation attacks resulting from the same identifier being used in multiple contexts (discussed in IoT-NGIN deliverable D5.4 [D5.4]). We also utilise another SSI-technology, Verifiable Credentials (VCs), to carry information about the trustworthiness of different parties (discussed in [D5.4]) and to implement decentralised access control solutions (Solution 7 detailed in Section 6.1). The use of DIDs and VCs has been previously explored mostly in the context of people and organisations, but here we are focusing on their use for (constrained) things (IoT devices) and the related twin, in order to bring the privacy and trust benefits also to this application area.

To make the use of Semantic and Digital Twins convenient, we are also exploring using digitally signed QR codes and GS1 Digital Links as a convenient and secure way to discover the Twins related to a particular IoT device (Solution 8) as detailed in Section 6.2. These types of new usability-oriented solutions are required to enable wide-scale usage of Twin-based solutions.


Finally, to illustrate how these solutions work synergistically, a demo of IoT device configuration is being developed in WP7 as detailed in Section 7. It will deploy Solutions 5-8 in the Jätkäsaari Living Lab to demonstrate how we can improve cybersecurity and protect users' privacy in an easy-to-use manner.

DRAFT - PENDING

3 Early attack detection and mitigation


This section summarises the motivations behind Solutions 1-4, provides an update on final versions of the solutions, and explains in more detail how the solutions collaborate in early attack detection and mitigation.

3.1 Malicious Attack Detector



IoT-NGIN introduces a number of cybersecurity tools that aim to shield Federated Learning systems from malevolent behaviours. These tools are designed to face common threats that are observed within the federated learning framework. IoT-NGIN has introduced the first version of the Malicious Attack Detector (MAD) in the deliverable D5.2. This detector is an advanced tool that can identify and block malicious activities within the federated learning system. In this document we introduce the label flipping mitigation technique that aims to eradicate the effects of such attacks. This technique relies on the assumption that the federated server has a small, clean dataset and can train the global model for a few rounds locally after the federated training process has ended. Together with the label flipping mitigation technique, MAD now provides an extra layer of protection against cyber threats, thereby enhancing the overall security of the federated learning environment.

3.1.1 Motivation for MAD



Federated learning systems are prone to several attacks due to their distributed nature. The attacks could, for instance, either alter the local data or the local model of a client to inject a specific pattern, degrade global model accuracy, or prevent the global model from converging etc. If successful, these attacks are catastrophic for a federated learning system and can completely alter its behaviour and performance. Also, vanilla federated learning setups don't include security measures against such attacks and are, therefore, deemed extremely vulnerable. Without proper security measures, the potential risks and negative impact of these attacks can quickly escalate. Thus, it is crucial to implement effective detection and mitigation strategies to safeguard the integrity and confidentiality of the data and to ensure the proper functioning of the federated learning system.

Even though there are numerous types of attacks within the federated learning framework, in this work we focus on *label-flipping attacks* launched from the clients. More specifically, we simulate the attacks described in section 5 "*Poisoning attacks in FL System*" of deliverable 5.2. There, the malicious clients train a GAN on the local dataset and then subsequently use it to produce synthetic images. These images are then labelled to facilitate each attack accordingly. Here we focus on 2 main types of label-flipping attacks: (i) *Model degradation attack* where the malicious clients aim to reduce the classification accuracy of the global model and (ii) *Targeted label attack* where the malicious clients aim to inject some diseased grape leaves as healthy into the global model.

Our earlier work, presented in D5.2, proved that Model degradation and Label flipping attacks can be successful in a classic federated learning setup, highlighting the need for effective countermeasures. To this end, we have experimented with various established federated learning techniques and assessed their robustness against these attacks. This allows us to identify approaches that can effectively mitigate the impact of label-flipping attacks. Furthermore, in the present document we introduce a simple yet effective additional

defence mechanism named label-flipping mitigation that can improve the alleviation of label-flipping attacks in federated learning systems. This mechanism can help to enhance the overall security of federated learning and reduce the potential for malicious attacks. By the proposed defense mechanism with the well-established federated learning defenses, we can help make federated learning more resilient against backdoor attacks and better protect the privacy of users' data.

3.1.2 Description of the MAD solution

In the deliverable D5.2 we introduced a machine learning-based system designed to detect and prevent cyber-attacks. The system uses a combination of supervised and unsupervised learning algorithms to analyse network traffic and identify potential malicious activities in real-time. The supervised learning component of Malicious Attack Detector (MAD) (solution 2) is trained on a dataset of known attacks, enabling it to recognize patterns and behaviours associated with specific types of attacks. The unsupervised learning component of MAD is used to identify new or unknown attacks by analysing the behaviour of network traffic and identifying anomalies. Once MAD identifies a potential attack, it can take action to prevent it, such as blocking traffic from the source IP address. The system is designed to be scalable and can be deployed on large networks to provide comprehensive protection against a wide range of cyber threats.

This subsection describes the technical details of the implementation for some commonly used federated learning defenses and the proposed mitigation. In more detail, we present the common federated defenses that are employed in the literature, details of our approach, used metrics and federated training setup.

GAN-driven federated attacks

In the previous deliverable D5.2 we introduced 2 novel GAN-based attacks using synthetic generated images. The malicious clients jointly trained a GAN to create realistic images and thereafter launched label flipping attacks. The goal was to confuse the global model resulting in general or specific misclassifications. The GAN model used for image generation is shown in Figure 3.1.

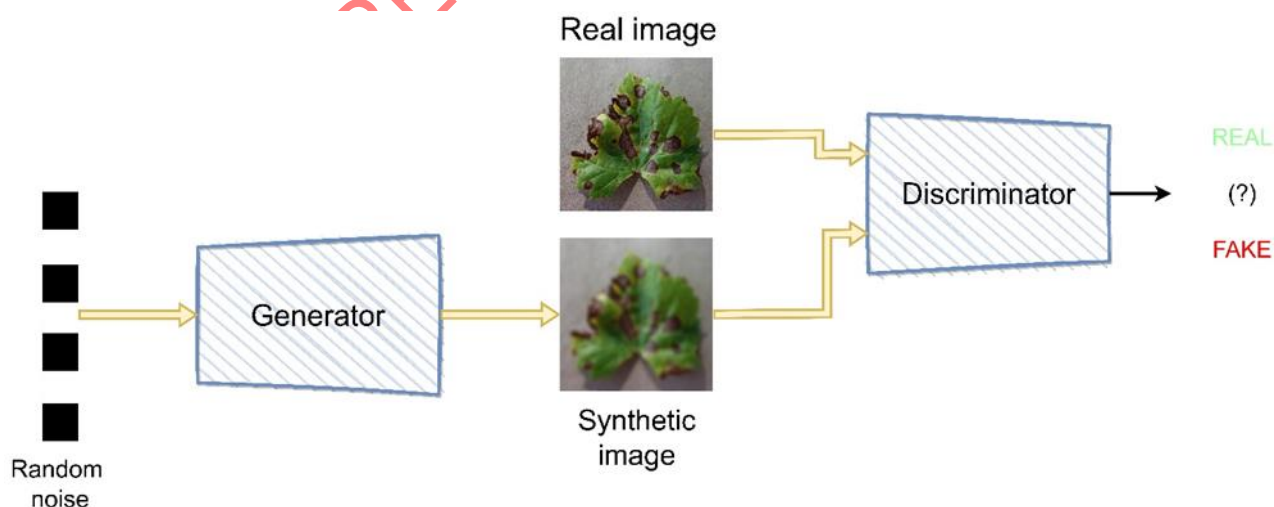


Figure 3.1 – GAN used for the Image generation of the malicious clients.

The dataset we used is publicly available and consists of grapevine leaf images with common diseases (e.g., Leaf blight, Esca)¹. Based on the local dataset of each client, the malicious clients were able to train this GAN to generate realistic grapevine leaf images. Consequently, these images were given wrong labels and were mixed with the local client dataset. This led to the deterioration of the global model's accuracy because it was trained on local datasets that have label-image inconsistencies. Experiments in our previous work demonstrate that these attacks are effective and achieve their corresponding goal resulting in a poisoned global model. Thus, these experiments indicate the need for a defence to filter the malicious updates and restore the performance of the model.

A detailed visual description of the whole federated learning process with the malicious/benign clients can be viewed on Figure 3.1.

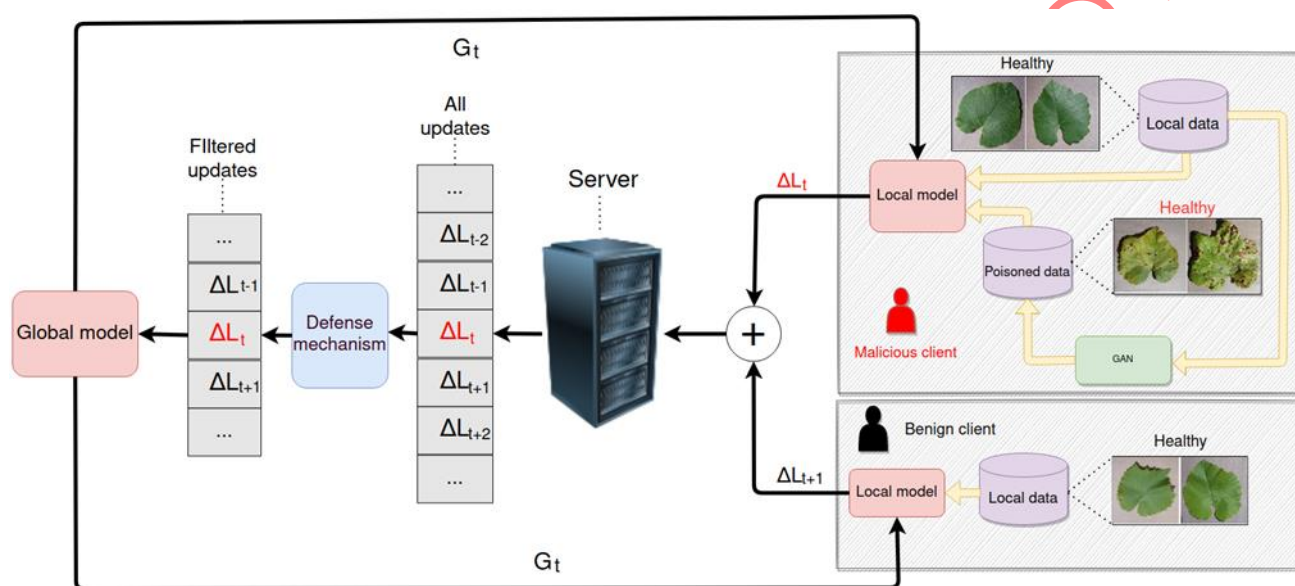


Figure 3.2 – Overview of the poisoning attack and aggregation mechanism within the federated learning framework.

Here, we see that a federated learning system can contain both benign and malicious clients. In our simulation, the malicious client uses the local dataset to produce a poisoned dataset, which is later added to the original local data. Implementing this strategy, the malevolent participant can inject the desired behaviour into the global model (e.g., misclassify healthy images as suffering from Leaf blight) through the synthetic images. We see that the federated server gathers all the updates from the clients and then using a defence mechanism attempts to filter the malevolent ones. Next, the aggregated model is given to the clients for the next federated learning round.

FL defences in MAD

To battle the aforementioned attacks, we validated a number of federated defences. These defences are variations of the aggregation rule and, in general, try to filter out malicious clients by viewing the model updates sent from the clients. The basic assumption is that the *model updates generated by malicious clients differ significantly* compared to the benign ones'. This difference can be investigated in the parameter space, and one can measure

¹ <https://data.mendeley.com/datasets/tywbtsjrjv/1>

the distance between model parameters separately or between whole models (the sum of model parameters). To measure this difference, common distances are the Euclidean distance, cosine distance, etc. Thus, we test our attacks against the defences of:

- Median aggregation
- Trimmed mean aggregation (with known number of malicious participants)
- Krum aggregation

The first two defences operate at the parameter level and treat each parameter separately. Thus, they can result in a partially poisoned global model where some parameters are aggregated using only benign updates and other parameters using both benign and malicious updates. On the other hand, Krum treats each model separately, and thus, in a federated round, it is possible for a completely poisoned local model to be selected as the global one.

Label flipping mitigation

Here we present a novel approach aimed at bolstering the defence against potential attacks in the federated learning system. Our method relies on the assumption that the server possesses a limited dataset consisting of pristine, labelled images, and can continue to train the model for a few more rounds after the client training phase. This technique serves to mitigate any malicious activity and rectify the model's performance. To elaborate further on the implementation details, we continue training the global model for an additional $R = 5$ rounds on the server following the conclusion of the federated learning process. This extended training period allows the server to refine the model's performance further, thereby minimising the impact of any potential attacks that may have occurred during the federated learning phase.

Metrics

The confusion matrix is a table used to evaluate the performance of a classification model by comparing the predicted and actual class labels. In the case of multiclass classification, the confusion matrix is a square matrix with dimensions equal to the number of classes. Each row in the matrix represents the instances in a predicted class, while each column represents the instances in an actual class.

The entries in the confusion matrix correspond to the number of instances that were classified correctly or incorrectly. The diagonal entries represent the number of instances that were classified correctly for each class, while the off-diagonal entries represent the misclassified instances. The sum of the entries in each row gives the total number of instances predicted in that class, while the sum of the entries in each column gives the total number of instances in the actual class.

An example of a confusion matrix for a classification of 4 classes can be seen on Table 3.1.

Table 3.1 – Confusion matrix outcomes of multiclass classification.

Classes	A	B	C	D
A	+correct values	-wrong values	-wrong values	-wrong values
B	-wrong values	+correct values	-wrong values	-wrong values
C	-wrong values	-wrong values	+correct values	-wrong values
D	-wrong values	-wrong values	-wrong values	+correct values

Multiclass classification accuracy is a metric used to evaluate the overall performance of a classification model with multiple output classes. It measures the portion of correctly classified instances out of all instances in the dataset. Mathematically, let TP , FP , TN , and FN be the number of true positives, false positives, true negatives, and false negatives, respectively. Let k be the number of output classes. The accuracy metric can be expressed as:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

where $TP + TN$ represents the total number of instances that were classified correctly, and $TP + FP + TN + FN$ represents the total number of instances in the dataset.

3.1.3 Results

This section presents a comprehensive overview of the experiments carried out to validate the proposed label-flipping mitigation approach. To assess the robustness of common federated learning aggregation algorithms, we evaluated the effectiveness of the two data poisoning attacks mentioned earlier. Specifically, we first test the secure aggregation techniques against these attacks and present differences compared to the federated average case. Next, we add the proposed solution on top of these defences and compare results.

In this section we describe the implementation details of the federated learning process. We created a classic federated scenario consisting of multiple clients and a server that operates as the aggregator. We also divided the clients into malicious and benign and assumed that the malicious participants collaborate and launch a concurrent attack. Moreover, each client (malicious or benign) has the same number of samples, which are all independent and identically distributed. We also use a simple convolutional neural network as the global model which is not pretrained. Parameters for this process can be seen in Table 3.2.

Table 3.2 – Parameters of the federated training process.

Parameter	Value
Learning rate	0.001
Local rounds	5
Federated rounds	50
Benign clients	70%
Malicious clients	30%
Batch size	32

Model degradation attack

In this case, a GAN is deployed on the malicious nodes with the intention of causing global model performance degradation. The adversary uses a GAN trained with all four classes. Using this model, the attacker generates samples of all four classes and later distributes these samples to the compromised nodes. Subsequently, each malicious node randomly assigns a class label to the generated images. Each local model will be trained on images that resemble the original ones but have random labels, which is something that confuses the classifier. It is noted that the poisoned dataset was created before the training process and is available at the beginning of the training procedure.

Experimental results evaluating the model degradation attack for the cases in which there is no defence or some of the three defence mechanisms can be seen in Figure 3.3. Firstly, the blue line shows the case of no attack, where the FL system contains only benign nodes, and the aggregation is a federated average to be used as a baseline. The green line shows the accuracy of the proposed attack for the case of federated average aggregation as the training procedure progresses. Regarding the other four classes, the only thing that changes is the aggregation algorithm. Orange is median aggregation, red is Krum aggregation, and purple is trimmed mean. Specifically, we can see that there is a difference of around 25-30% in accuracy between the case in which no attack is realised and the case in which an attack has materialised with no defence in place. Moreover, regarding the defences, the model accuracy is improved when some defence mechanisms are adopted compared to the case of no defences. However, the improvement is approximately 5, 7, 5 percentage points for the median, Krum, and trimmed mean approaches, respectively. The low improvement in all three cases implies that these mechanisms fail to effectively identify the malicious nodes and that the poisoned updates are significantly infiltrating the global model. It is worth noting that in the case of the trimmed mean defence, we have assumed that the parameter n (number of malicious clients) is known beforehand, and thus, the defence should be more powerful.

Nevertheless, all three approaches consistently achieve an accuracy value below that of the global model, with the difference ranging from 5 to 20 percentage points. We thus assume that this kind of attack is stealthy mostly because the poisoning procedure closely resembles the training procedure and that the malicious nodes have both a poisoned and a clean dataset.

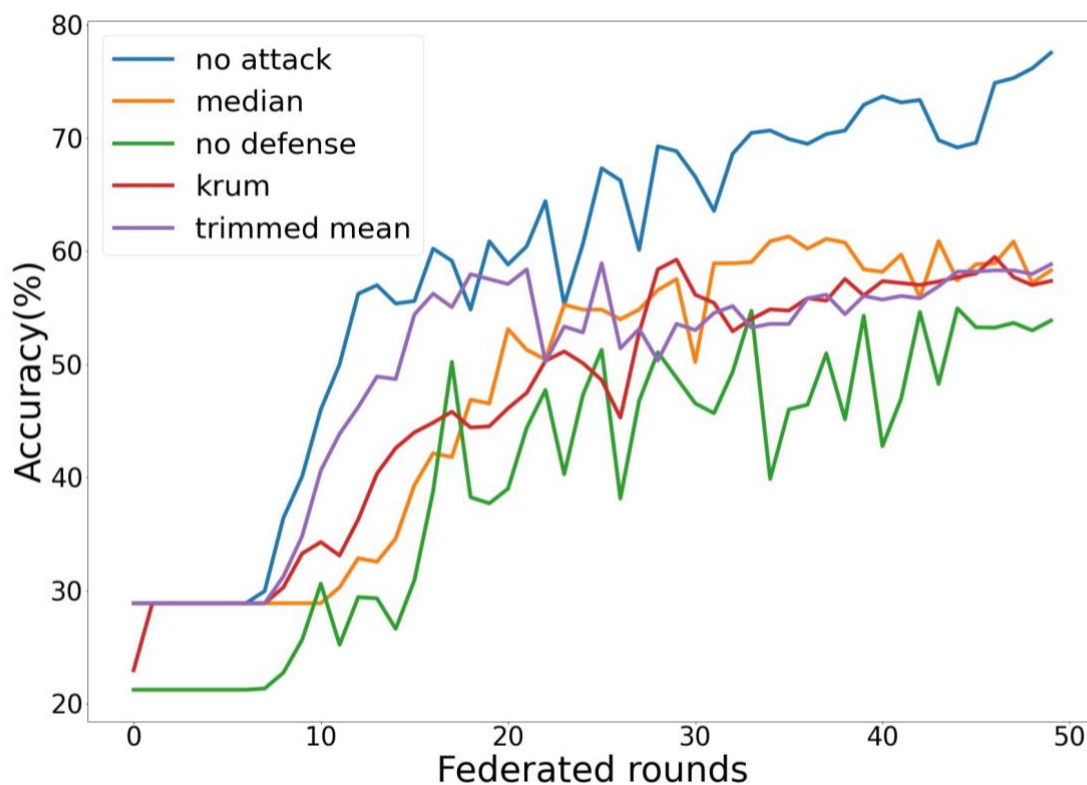


Figure 3.3 – Accuracy of the classification model with different defence mechanisms.

Targeted label attack

In this case the test scenario includes 70% benign clients and 30% malicious. All benign clients have the same amount of data (images). The malicious clients generate images of a specific disease class, namely the leaf blight class, and then flip the label of the synthetic images to the one corresponding to healthy ones. The goal is to trick the model into classifying leaf blight images as healthy. This is indeed a targeted label attack, and the results could be catastrophic for a classification model.

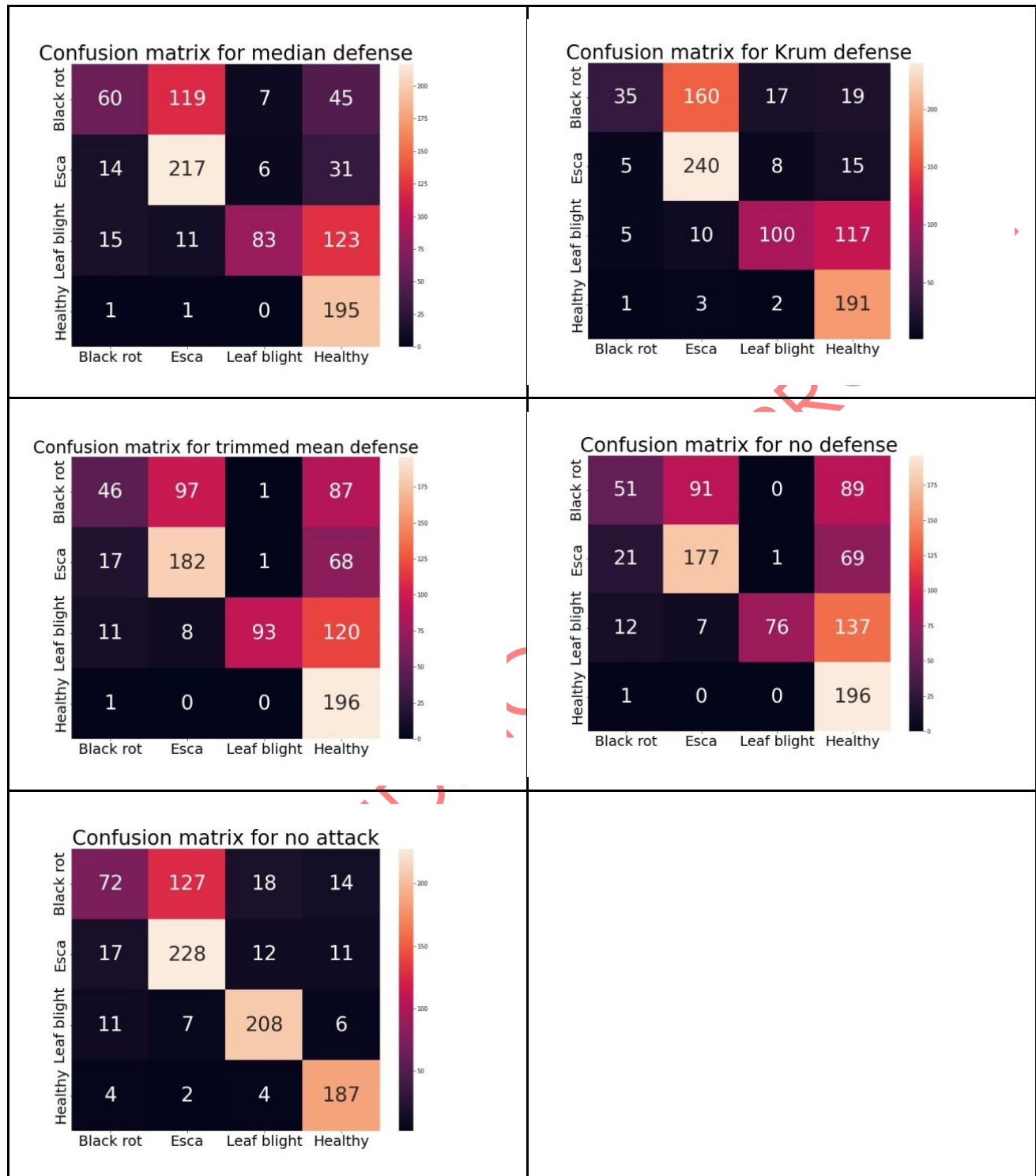


Figure 3.4 – Confusion matrices for the targeted label attack with different defences.

Results of the experiments conducted for evaluating the targeted label attack with the three or no defence mechanisms adopted are depicted in Figure 3.4. First, comparing the case with no defences to the case in which there is no attack, the attack appears to be successful for both the tasks of stealth and targeted label attacking. Specifically, this attack results in a model that greatly misclassifies leaf blight as healthy, but achieves similar accuracy

regarding the classification task for the other classes. In particular, 58% of leaf blight images have been categorised as healthy compared to 2.5% for the case of no attack. Regarding the effectiveness of the defences, they slightly improve the accuracy of this specific label. Specifically, with any of the defence mechanisms applied, the misclassification of leaf blight as healthy is approximately 51%. This happens since these methods are performed at the parameter level and not for the whole model. This happens because a specific set of parameters (e.g., biases of the last layer) may be easily identified as poisoned by the defences. However, viewing the results, we can assume that this set is small compared to the total number of parameters, and the bulk of the poisoned parameters are incorporated into the global model. Hence, it is obvious that these methods fail to select only benign client models, and the attack is deemed a success.

Mitigation

The results of the proposed mitigation approach are presented in Figure 3.5 the targeted label attack and Table 3.3 concerning the model degradation attack.



Figure 3.5 – Confusion Matrix after applying the proposed mitigation technique.

In the case of the targeted label attack, the method successfully mitigates the effects of the attack, as evidenced by the model no longer misclassifying leaf blight images as healthy. Additionally, the results for the other classes remain unchanged, indicating that the attack reduces the accuracy of different classes, and the attack is indeed targeted, as the accuracy of other labels remains relatively stable.

Table 3.3 – Test set accuracy for the case of with/without mitigation.

Mitigation	Accuracy
yes	62%
no	50%

In the case of the model degradation attack, the proposed method quickly alleviates the effects of the attack, with the model achieving the accuracy of a non-attacked model within only five additional rounds. This result represents a substantial improvement of approximately 12%. Overall, these findings demonstrate the effectiveness of the proposed

method in mitigating backdoor attacks while remaining simple and easy to implement. However, it is important to note that the approach relies on the assumption that the central server has access to a small dataset of clean labels, which may not always be feasible in real-world applications.

3.2 Moving Target Defense Honeypots

Moving Target Defense (MTD) Honeypots (solution 4) is an additional defence mechanism that was developed in order to alleviate an increasing number of threats that target IoT and FL systems and ecosystems.

3.2.1 Motivation for MTD Honeypots

MTD Honeypots framework constitutes a honeypot-based technical solution that also incorporates dynamic configuration capabilities. The term “Moving Target Defense” suggests the dynamic shifting of a system's exposed resources in a continuous manner. In the IoT-NGIN project, the MTD term refers to the ability of the framework to alter the provided functionality and network configuration based on some input.

The abovementioned capability is considered an effective countermeasure against contemporary cyberattacks. This is because the honeypots that are deployed in an IoT network are constantly changing their associated network and system configurations, which in turn makes the established honeypots hard to be detected by an adversary. At the same time, the MTD Honeypots framework succeeds in decreasing the attackers' knowledge over an IoT system.

Furthermore, Honeypots is a cybersecurity solution that capitalises on a manufactured attack surface which exposes a specific set of vulnerabilities that aim to attract cybercriminals' attention. These measures allow for the legitimate targets to escape some of the adversarial attention. In addition, Honeypots can also be utilised for gathering intelligence about the identity, motivation and relevant attack patterns used by the attackers.

MTD Honeypot framework is designed to work with the Vulnerability Crawler (VC) (solution 3) which is able to scan IoT networks and provide the results to the MTD Honeypots solution in order to drive the deployment of the most appropriate, and thus, effective honeypots.

3.2.2 Description of the MTD Honeypots solution

IoT-NGIN developed Honeypots framework utilises the IP randomization MTD technique in order to change the network properties of the attack surface exposed by the deployed honeypots. This section provides a summary of MTD Honeypots technical design as the detailed description was already provided in D5.2. Moreover, this section contains the descriptions of the additional updates that were applied to the developed framework. In short, this includes the incorporation of additional Honeypot tools and the creation of a HELM Chart that provides for easy installation. The high-level overview of the MTD Honeypots framework is presented in figure 3.6 below.

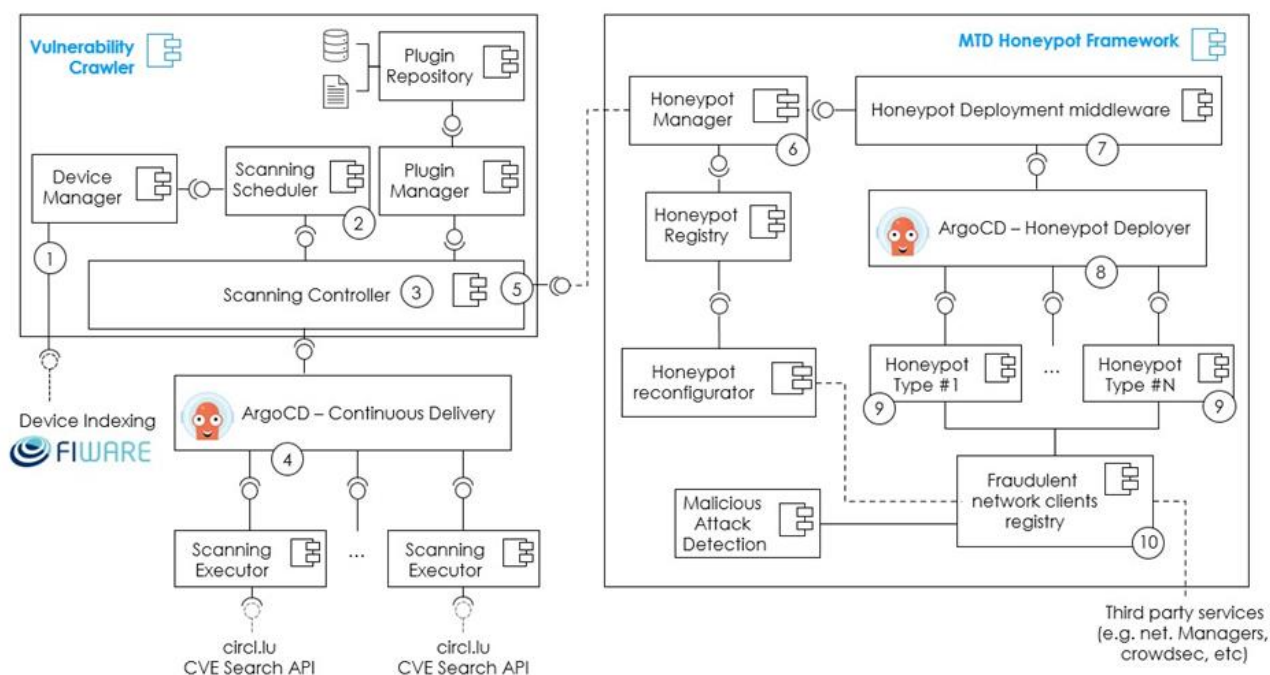


Figure 3.6 – IoT-NGIN MTD Honeypot framework overview.

The heart of the developed MTD Honeypots framework is the Honeypot Manager (HM). HM interfaces with the VC and periodically queries for vulnerabilities that are identified in the target system. Based on this information the HM then selects the most appropriate honeypots that will be deployed in the IoT network. In addition, the volume of the vulnerabilities that have been detected by the VC suggest the number of honeypots' instances that should be deployed in order to create a strong enough attack surface.

The available Honeypot solutions are stored in the Honeypot Registry (HR). HR contains the list of the available honeypots and the associated meta-data that describes the set of vulnerabilities that can be exposed by each honeypot. Initially a limited number of honeypot solutions were included in the HR, namely Cowries and Log4Pot. However, as the development proceeded we opted to extend the available honeypot tools and to include two additional honeypot solutions, the ddosspot and Dionaea.

- Cowrie² is a versatile honeypot and can be considered either as medium or as a high interaction SSH and Telnet honeypot designed to log brute force attacks and the shell interaction performed by the attacker.
- Log4Pot is³ a honeypot that can detect the Log4Shell vulnerability (CVE-2021-44228). Log4Pot is able to listen on various ports for Log4Shell exploitation and detect exploitation requests observed in the request line and headers.

² <https://github.com/cowrie/cowrie>

³ <https://github.com/thomaspatzke/Log4Pot>

- DDoSPot⁴ is a honeypot "platform" for tracking and monitoring UDP-based Distributed Denial of Service (DDoS) attacks.
- Dionaea⁵ is a low-interaction honeypot that captures attack payloads and malware. Dionaea is meant to be a nepenthes successor, embedding python as scripting language, using libemu to detect shellcodes, supporting IPv6 and TLS.

Honeypot Deployer (HD) collects this information and subsequently creates the required workflows that aim to deploy and configure the necessary honeypots. HD was developed capitalising on the ArgoWorkflows CD framework.

Then, the necessary honeypots are deployed and the Honeypot Logs Collector (HLC) is responsible for collecting the logs that are generated by the honeypot instances. The logs capture information that describe the adversaries' activity within the honeypots and also include the attackers' connection details. These logs are stored in a database.

Finally, the Honeypot IP Randomization (HIPR) module facilitates the enforcement of the IP randomization technique for the deployed honeypots. Once the initial honeypot instances are deployed, they get a random IP address assigned to each one of them. This IP address is pulled by the available IP pool that the Kubernetes cluster uses for the deployed resources. HIPR is a job scheduler which works closely with Kubernetes Load Balancer. The purpose for this tool is to periodically execute jobs that change the IP addresses of the deployed honeypot instances.

This MTD technique effectively prevents the attacker from identifying the honeypots, and thus, significantly lowers the possibility of the adversary to blacklist the exposed IPs that are assigned to the honeypots, keeping the legitimate part of the IoT network secure.

The installation guidelines for successfully deploying the IoT-NGIN MTD Honeypots framework have been described in D5.2 section 8.3. The source code of the MTD Honeypots framework is available through the GitLab repository of the project⁶. Moreover, a HELM Chart has been defined for the Honeypot framework, which greatly simplifies the deployment process. The corresponding installation steps are described within the Readme that resides in the aforementioned repository.

In this section we demonstrated the developed mitigation method that was able to mitigate common federated learning GAN-based attacks. This method relied on the assumption of having a small dataset on the aggregator which could be used to retrain the global model after the federated learning training had completed. Results showed that our methodology was able to remedy the results of the attacks achieving a 12% boost in accuracy. Moreover, we have completed the presentation of the early attack detection and mitigation cybersecurity tools by incorporating an overview of the MTD Honeypots design and implementation, highlighting also the tool's latest technical updates.

⁴ <https://github.com/aeth/ddosspot>

⁵ <https://github.com/DinoTools/dionaea>

⁶ <https://gitlab.com/h2020-iot-ngin/enhancing-iot-cybersecurity-and-data-privacy/mtd-honeypot-framework>

4 Semantic Twins

This section describes the Semantic Twin (ST) solution, which is used to describe the Real-world entities (e.g. IoT devices) and their Digital Twins (DTs) in a machine-readable manner as shown in Figure 4.1. The following subsections describe the motivation for building Semantic Twins and the details of the Semantic Twin solution.

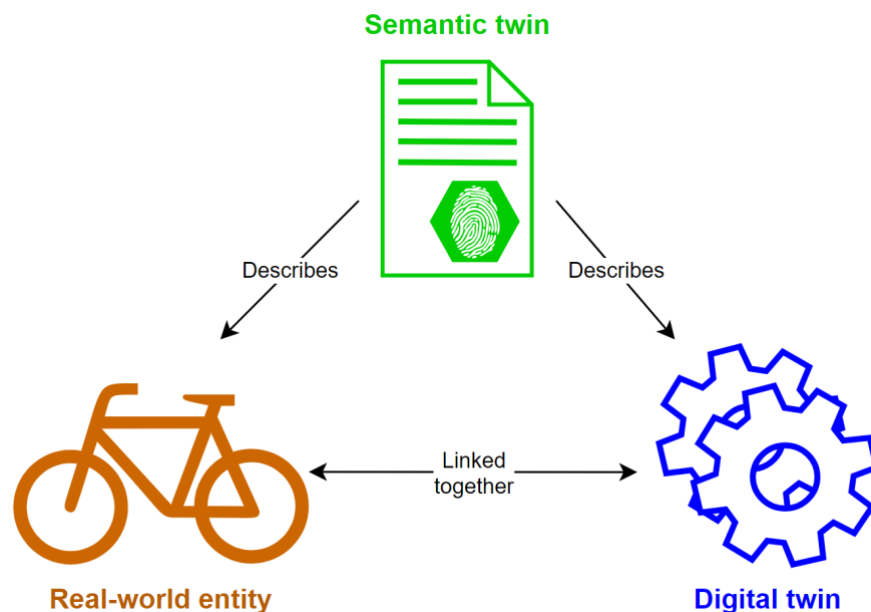


Figure 4.1 – A Semantic Twin describes a real-world entity and its Digital Twin forming an (entity) Triplet. The real-world entity may be a physical device, but also a more abstract real-world entity, such as an organisation.

4.1. Motivation for Semantic Twins

Recent years have brought us smart entities that consist of a physical entity and its Digital Twin (DT). However, DTs are currently not defined well enough to easily build scalable applications on top of them. Legacy Digital Twins are also missing the basic components needed for data privacy and trust, something the IoT devices themselves also often crave.

The following subsections discuss issues in IoT systems and Digital Twins, lay out requirements for Semantic Twins, and describe the role of Semantic Twins.

4.1.1 Issues in IoT systems and Digital Twins

Legacy IoT devices are configured in a myriad of ways. While this approach has worked well for isolated use cases, it has not enabled IoT devices to act in a properly networked manner. Three important root causes are:

- IoT devices are (in most cases) constrained in technical capabilities (e.g. limited computation capability, communication bandwidth, and power resources).
- IoT devices require a high degree of security and trustworthiness due to being able to create damage in the real world.

- The lack of scalable technical solutions for traversing between the physical and digital worlds (e.g. conveniently accessing sensor data while being physically close to the sensor).

Most of the technical constraints can be overcome with the usage of Digital Twin solutions, but achieving an adequate level of security and trustworthiness in a networked environment still requires new solutions. Digital identity solutions can be leveraged to solve some of the trustworthiness issues, but others need further types of arrangements, such as suitable data management architectures.

Digital Twins are virtual counterparts of real-world things. From there on, the definitions diverge according to the underlying use case. The Digital Twin concept originated from the product lifecycle management domain in engineering and was adopted as a metaphor for a simulation model that is connected to a real-world machine. Simultaneously, the IoT domain developed concepts and solutions, such as digital agents and sensing technologies, that would later be integrated into the Digital Twin concept. Furthermore, many other digital technologies such as artificial intelligence and augmented reality have been associated with Digital Twins, making the concept fruitful ground for misunderstandings.

For the purpose of this document, we define a Digital Twin as a *collection of software services that are related to a real-world entity*. Some of the software services may be accessible through the public internet, others only in an isolated network and running on local machines. All of these services may provide value for people dealing with the corresponding real-world entity, but there are no conventions on how to deal with these heterogeneous solutions.

4.1.2 The role of Semantic Twins

Semantic Twins are being developed in the IoT-NGIN project as a general solution for adding metadata to Digital Twins and real-world entities. While Digital Twins are complex digital services that can accomplish almost any digital task, Semantic Twins give context and meaning to Digital Twins and real-world entities by providing information about the services of real-world entities and their Digital Twins in a unified human and machine-readable format.

Digital Twins consist of digital services that are related to the real-world entity. These services can be very diverse, such as a cloud-based IoT platform, simulation model, database, or an artificial intelligence agent. These services are also implemented in diverse ways and may be accessible in the cloud or only as local software that is run without internet access. The Semantic Twin needs to be able to provide its services in all of these situations.

Semantic Twin is a solution for improving interoperability and they aim to make the integration of Digital Twins and their real-world counterparts more structured and efficient. To achieve this goal, Semantic Twins consist of three main components as shown in Figure 4.2: Twin ID, twin document, and semantic descriptions, which are further detailed in section 4.3. The Twin ID enables the identification of the Semantic Twin, and therefore, the ST-DT-entity triplet, and this identification can be linked to the real-world entity and Digital Twin services through the descriptions. For example, an external service may access the database service of a Digital Twin via the Twin ID and the semantic description of that service.

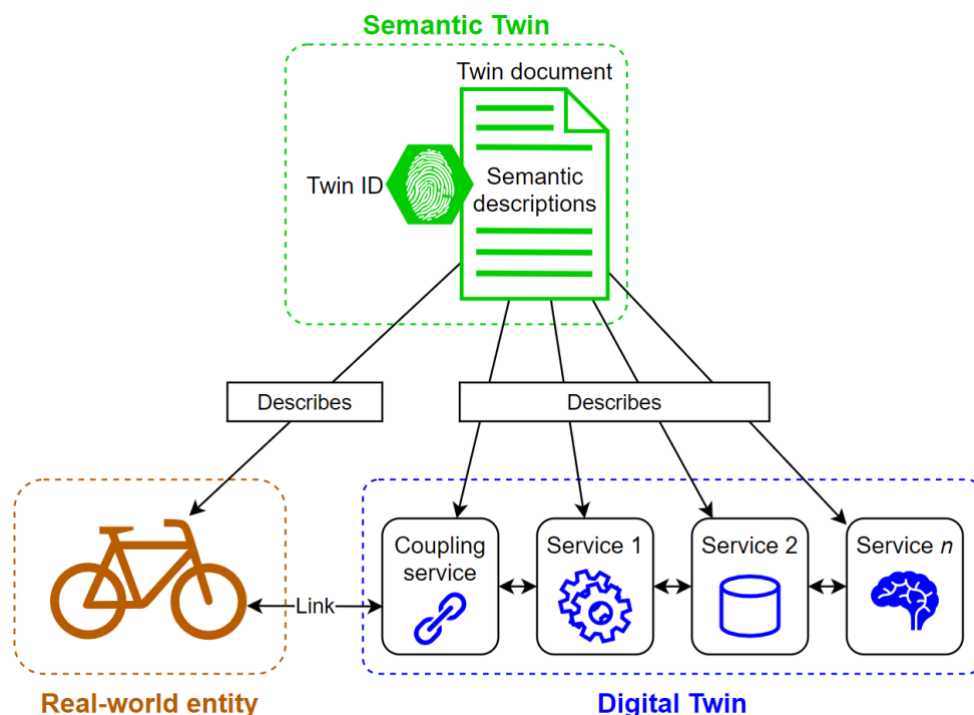


Figure 4.2 - A detailed look into an entity triplet, showing the basic features of a Semantic Twin and how it describes the Real-world entity and the Digital Twin.

As an example case, twin documents have been used in machine-to-machine communication in a simulated factory, where machines accessed the communication details and relationship descriptions of other machines from their twin documents to fulfil a logistics-related task [Mat2022]. This approach, however, assumes that all parties are trusted, limiting its applicability only to environments to where access is restricted from the outside. In the long term, Semantic Twins can help create a global network of Digital Twins. We call this network the *Digital Twin Web* due to the intended analogy to the World Wide Web as further explained in [Aut2021a]. However, in this wider open environment, the related trust and privacy issues require more advanced solutions as discussed later in this document.

4.2 Description of the Semantic Twin solution

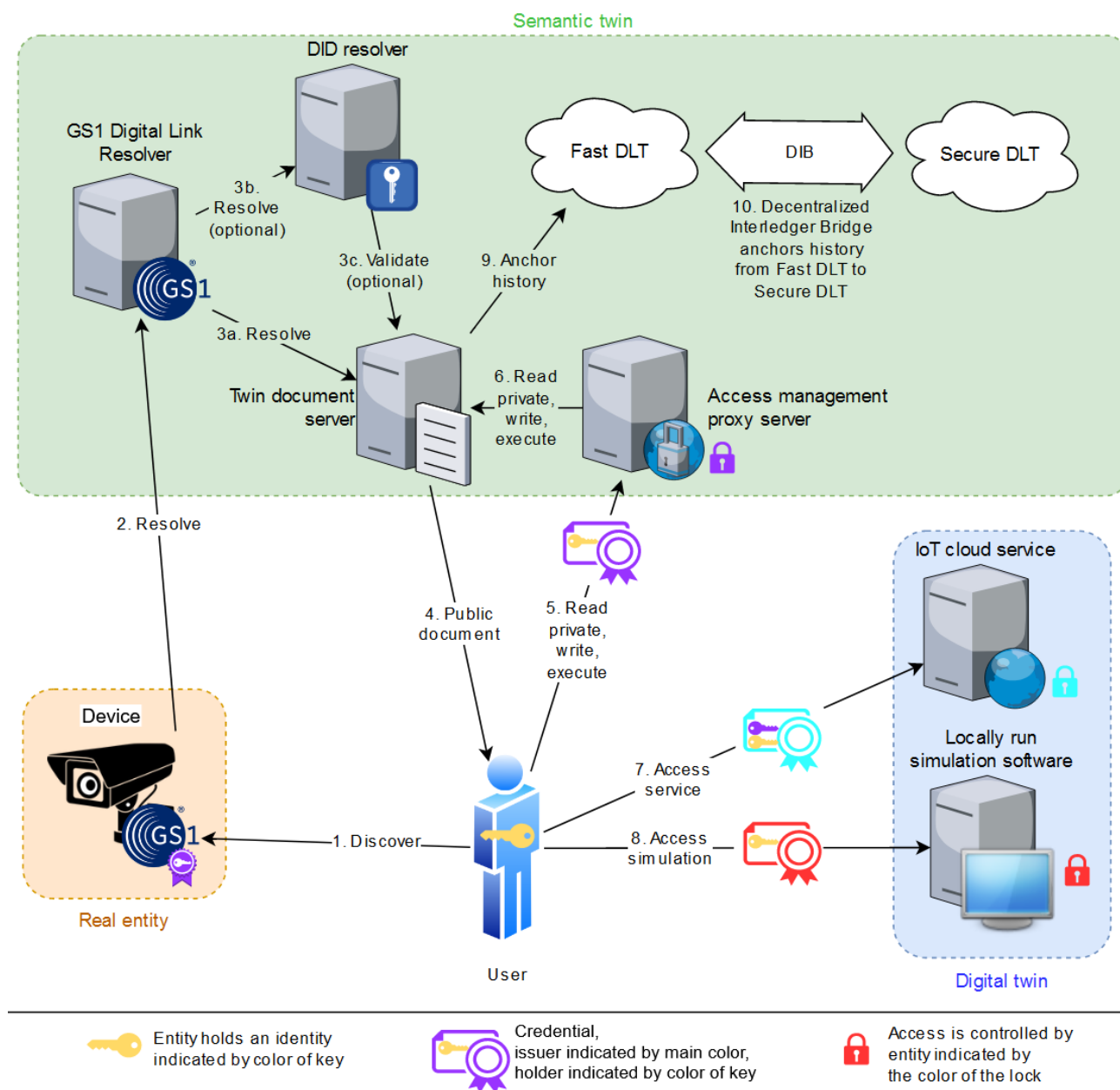


Figure 4.3 – Example composition of a system of systems that uses different features of Semantic Twins.

The functional architecture of a system that uses the Semantic Twin solution is shown in Figure 4.3. The twin document is the central component of the Semantic Twin, providing the main body of information. Other components in the green box provide various services for enhancing discoverability and trustworthiness of the solution.

From the user perspective, the Semantic Twin journey starts from (1) the discovery of an identifier, which in this case is the GS1 Digital Link. It can be discovered via a QR code on the physical device or as a text string around the internet. The GS1 Digital Link (2) resolves via the

Domain Name System (DNS) to a GS1 Digital Link Resolver, which (3a) by default resolves to the twin document server, but may also (3b) resolve to a DID resolver when read with specially made software. The DID can then (3c) provide additional validation for the twin document. (4) The public part of the twin document is then sent to the user. If the user holds the appropriate credentials, they can also (5) read the private part of the twin document and modify it, and execute operations via an access management proxy server that (6) redirects the requests to the twin document.

The user reads the twin document that describes the methods to access an IoT cloud service and a locally run simulation software. The user decides to (7) access the IoT cloud service with a (delegated) credential. Then, the user (8) accesses a simulation in a local environment with another credential that requires no internet access.

To have a shared immutable history for the twin documents, (9) the twin documents are hashed and the hash is stored in a fast distributed ledger, which already provides a good level of confidence on the history. (10) At longer time intervals, the hashes are then grouped and stored (with salt to preserve privacy as discussed in Section 4.1.2) to a more secure ledger via a Decentralised Interledger Bridge (DIB) to further increase the confidence through this more secure ledger.

As demonstrated by the description of the architecture, the main services of the Semantic Twin solution are:

- Provide a description document of the real entity and its Digital Twin services.
- Provide a resolvable ID for the entity triplet.
- Provide validation of the twin document.
- Manage access to the document and potentially to the device and Digital Twin.
- Verify the history of the twin document, in both fast and secure methods.

The three main topics, twin document, discoverability and trustworthiness, and semantic descriptions, of the Semantic Twin solution are further described in the following subsections.

4.2.3 Twin document

A twin document (Digital Twin description document) is a text document that describes a Digital Twin and its real-world counterpart. A twin document is supposed to be the initial source of information about a real-world entity in all use cases. As the document is text-based, any dynamic materials are added as links or interface descriptions.

The distinction between a twin document and a semantic description is that a twin document provides the overall format, and semantic descriptions are the actual contents written in that format. Hence, a twin document is kind of a shell for more detailed information.

We currently use unstandardized formats for twin documents because no standard format fulfils enough requirements to be useful enough for the intended purposes. Unstandardized formats can be used in limited experiments and applications, but in the long term, a standardised format is required to achieve most of the benefits of Semantic Twins. Currently, the strongest candidates for twin document format are:

- *Asset Administration Shell (AAS) [AAS]*

- *Web of Things Thing Description (WoT-TD)* [WoT-TD]
- *Digital Twin Definition Language (DTDL)* [DTDL]
- *Next Generation Service Interfaces-Linked Data API (NGSI-LD)* [NGSI-LD]

Those were compared by [Jac2020]. We currently balance between the solutions, but have decided to use JSON-LD as the format of our twin documents. We added support for JSON-LD to the open-source twin document hosting software “Twinbase” [Twinbase] and developed the Semantic Twin ontology in a format that supports JSON-LD. The general concept of the twin document was introduced by [Ala2021] and a method to distribute them was introduced by [Aut2021b]. To be clear, the term *twin document* refers to the overall concept and not any specific style of implementation.

4.2.2 Discoverability and trustworthiness

The discoverability and trustworthiness of Semantic Twins are achieved with various identifier/identity and distributed ledger solutions. Discoverability is implemented with a “Twin ID” concept, whereas trustworthiness is a more complex combination of Twin ID and other solutions such as Verifiable Credentials (VCs) for third-party information, with a special focus on distributed ledgers to provide an immutable history for the ST.

The term “Twin ID” refers both to the identifier and identity solutions of Semantic Twin systems. We use both of these terms because they are conceptually different and have different technical implementations, but still either of those might be needed depending on the use case. Some Semantic Twin use cases may require a full-fledged identity solution with advanced features, such as verifiable credentials, whereas other cases might require anonymity and therefore use temporary identifiers for privacy reasons. Also, depending on the use case, separate IDs may be needed for each Digital Twin service as well as the real-world entity. In addition to one-way linking from a Semantic Twin to a Digital Twin, it may also be beneficial to implement a bidirectional linking. For example, a Digital Twin service may update its own description in the Semantic Twin to keep it up to date, or a Digital Twin service may use the credentials administered by the Semantic Twin to access restricted information in other Digital Twins.

Three main methods of Twin IDs have been identified: a plain URL, a GS1 Digital Link, and DIDs of different methods. Twin ID technologies are still under development, and we use simplified solutions to get started immediately. A baseline solution for a Twin ID is to use a *dedicated URL* as an identifier for a twin so that the URL is redirected to the corresponding twin document. This however does not allow more granular features that the use of GS1 Digital Links and DIDs enable. *GS1 Digital Links* enable several redirects from one URL, and *DIDs* enable e.g. short-lived identifiers and the assignment of a verifiable credential that can be used to access various services. However, simple URL redirections are readily available on the internet for free, whereas GS1 Digital Links require hosting or paying for a server, and DIDs require that the user holds and uses cryptographic keys correctly. These may not be obstacles for organisations with strong research and development capabilities, but may hinder at least short-term adoption in more production-oriented organisations. The initial versions of the Twin ID concept and the Digital Twin identifier registry along with their initial PoC implementations with URLs were described by [Aut2021b].

Trustworthiness can be increased via Twin ID solutions on various levels. Trusting a plain URL or GS1 Digital Link requires that the DNS itself and the holder of the domain are trustworthy.

Additional trust can be established by signing URLs or documents with DIDs, although this requires that the user knows and trusts the signer. DIDs can also create decentralised chains of trust through the use of verifiable credentials. The chains can be used e.g. for delegating access management rights to a system through a chain of organisations. The VCs can also be used to issue trustworthy third-party statements about the ST, DT and/or the real-world entity. For instance, a certification authority can issue a statement that a specific IoT device has been correctly installed and calibrated, and therefore, produces trustworthy data.

Distributed Ledger Technologies (DLTs) can be used to provide immutable history for twin documents by storing each version of the twin document to the DLT. This makes it easy to check the contents of each version and when they were created simply from the DLT. However, a twin document can be quite large, so storing the whole document to the DLT is not necessary for the immutable history, we can simply calculate a hash of each version and store only the hash. This saves a lot of space and costs on the DLT, but requires the twin documents themselves to remain available elsewhere for verification. An additional benefit of only storing a hash is that the contents of the twin document are not exposed publicly, but the existence of the hash at a certain time in the ledger proves that the twin document (with that specific contents) existed at that point of time. The trustworthiness of the DLTs varies and usually correlates with the costs of using that ledger: a lower degree of trustworthiness can be achieved by storing the hash to a fast and cheap ledger, whereas a high degree of trustworthiness requires storing the hashes to a globally known secure (and usually also expensive) ledger. A convenient solution is to leverage an interledger solution to achieve a high level of trust while keeping the cost low, as detailed in Section 5. There we can also use a nonce (salt) in the hashing process to prevent tracking of twin documents across ledgers for improved privacy.

4.2.4 Semantic descriptions

Semantic descriptions are the contents of twin documents. The use of globally shared ontologies makes the twin documents machine-readable across different implementations and is a key factor in enabling interoperability of real-world data across services.

For example, a visualisation software for city data can fetch the details of a data interface of a sensor device via a Semantic Twin, so that the user only needs to insert the Twin ID of the sensor to the visualisation software. Thanks to the semantic descriptions, the software will know the type of the sensor and visualise it in the correct way: a radar will be shown as a radar in the correct location and the observations of the radar will be included in the visualisation automatically.

A problem with using globally accepted ontologies for the semantic descriptions of twin documents in practice is that they do not currently cover enough use cases with high enough precision. Ontologies may also be difficult to find and many of them are not documented in a way that would enable fast adoption by people who are not deeply familiar with the conventions of the semantics field. In some cases, it may be necessary to create a new ontology, but the creation and publishing of them requires even more profound understanding of the conventions. We attempt to ease the barrier for adoption by introducing an ontology dedicated to Semantic Twins, described in the next section.

4.2.5 SSI API to Semantic Twin

To prove the technical feasibility of the overall Semantic Twin approach, we implemented the *access management proxy server* (the IAA proxy detailed in Section 6.1) shown in Figure 4.3. We use a research originated open-source software *Twinbase* [Twinbase] as a *twin document server*. However, the only method to modify twin documents in the original version of Twinbase is by Git operations. Hence, we developed first a wrapper server that transforms HTTP-based CRUD (Create, Read, Update, Delete) requests to Git operations. That wrapper server however needs an access management solution, for which we leveraged the access control proxy from T5.4 that allows us to use delegated VCs for access management. We call the combination of these two servers the *Twinbase SSI API* as shown in Figure 4.4. The Twinbase SSI API is the implementation of the conceptual SSI API to Semantic Twin. The source code for Twinbase SSI API can be found in GitHub [TB-SSI-API].

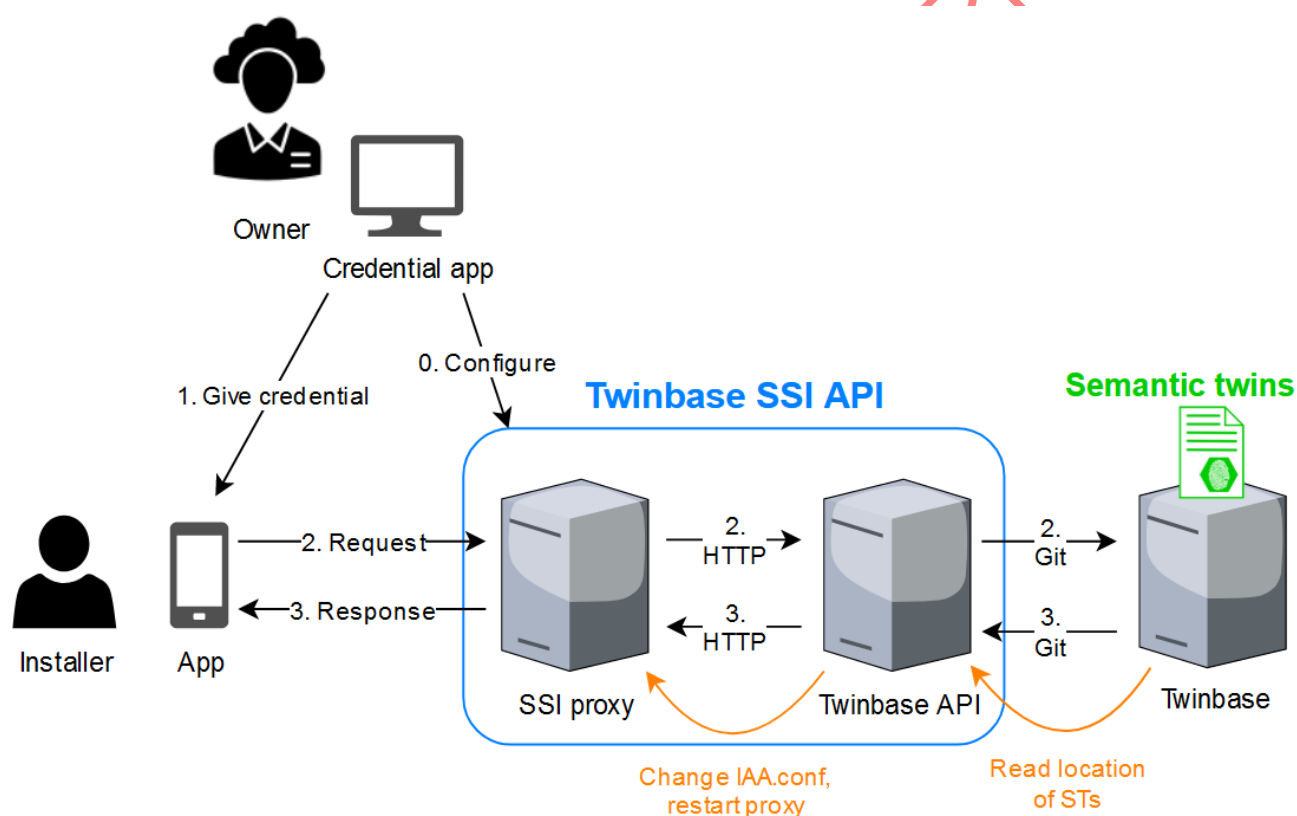


Figure 4.4 – Technical implementation of SSI API for Semantic Twins. Orange arrows depict the process of updating access requirements.

The process of using SSI API to Semantic Twin is as follows.

- 0. Owner of a semantic twin configures the SSI API for a selected semantic twin hosting server.
- 1. Owner gives credentials to an installer technician to access and/or modify the semantic twin,

- 2. The installer sends a CRUD request to the SSI API, which forwards the request to the semantic twin.
- 3. The semantic twin returns the results to SSI API, which forwards them to the installer.

To allow a user friendly and efficient process, a *credential app* and *installer app* should be used when communicating with the SSI API. These apps are, however, use case specific and were not developed as part of the semantic twin solution. However, a demo installer app is being prepared as part of the Smart City LL in WP7.

The SSI API dynamically determines the required user privileges according to information found from the semantic twins. In the current implementation, the SSI API reads the semantic twin document to check if the Neighbourhood⁷ class of SAREF extension for Smart City ontology [SAREF4CITY] is defined under the location⁸ class of WGS84 Geo Positioning vocabulary [WGS84voc]. If a neighbourhood is found, the SSI API mandates that anyone who wants to access that semantic twin has to hold a credential to access that specific neighbourhood.

The motivation for the SSI API comes from the *IoT devices configuration demo* described in Section 7.1, although the SSI API also works as a general method for privacy-preserving access control for semantic twins in any use case, where the owner and user can use apps that handle the use of identities and credentials.

4.2.6 Semantic Twin DLT integration

To anchor the history of semantic twins in a reliable way, the semantic twin solution was integrated to a DLT. To preserve privacy while taking advantage of the reliability of a public ledger, only hashes are stored in the ledger. This approach also lowers the costs of using the ledger.

The DLT integration was implemented to the Twinbase software. Twinbase generates hashes for the twin documents and pushes the hashes to a ledger. The Twinbase user interface also provides a button to verify if the document has been anchored in a ledger, showing the name of the target ledger and the time when the hash was stored.



Figure 4.5 – User interface showing the button to validate a document (a) and the result of the validation (b).

⁷ <https://saref.etsi.org/saref4city/Neighbourhood>

⁸ http://www.w3.org/2003/01/geo/wgs84_pos#location

The source code of the Twinbase with DLT integration can be found on GitHub [TwinbaDLT].

Instead of storing the hash directly to a public ledger, which may be so expensive that the benefit does not outweigh the costs, the hash of multiple twin documents may also be stored in a smart contract in a private ledger, which at suitable intervals combines the hashes and sends an event to an interledger, which transfers the hash to a smart contract in a public ledger. This public ledger can be used to validate a single twin document. This way, the user can enjoy the cost-efficiency of a private ledger while also taking advantage of the reliability of a public ledger where necessary.

The DLT integration for semantic twins is, in addition to the general benefits of anchoring the history, motivated by the Smart Factory use case. It is typical that powertrain operators need to make some configuration changes for powertrain components. For example, some parameter changes are typical when a process is to be optimised. Normally these kinds of changes mean behavioural changes in powertrain operation. It is obvious that there should be a way how these changes will be implemented in the digital version of the product as well. In some cases digital twins are used to simulate behaviours of real world entities.

The proposal for a regulation on cybersecurity requirements for products with digital elements, known as the Cyber Resilience Act, requires that mentioned changes in digital twin documents must be documented in the proper way [Cyber2022]. The idea is that all the changes can be tracked from semantic twin document history, and the operator can check what software version was used and what was the parameter setting.

4.3 The twinschema.org Semantic Twin ontology

One primary goal for the Semantic Twins is to enhance interoperability in the domain of Digital Twins. Semantic technologies are a key enabler for interoperability of heterogeneous data and information exchange. This goal is achieved by creating a specialised ontology for the domain of semantic twins. Semantic technologies and ontologies were already introduced in [D5.3] and [D5.4]. Therefore, only a brief summary is given here.

The "Ontologies" and "Vocabularies" are exchangeable in this context and can be defined as follows: *Vocabularies define the concepts and relationships (also referred to as "terms") used to describe and represent an area of concern* [Ont2022]. Ontologies focus on providing the vocabulary and the relations for a domain. They seldom incorporate information about individuals and instances of the classes, unless these incorporate some fundamental truth in the respective domain. Thus, the primary use case of an ontology is to provide knowledge about the relations and to define annotation points for actual data. The latter is then known as "linked data", and is for instance used by websites to give search engines more in-depth knowledge about their content.

Semantic Twins build upon this concept by using an ontology created for semantic twins to provide meaning to the data in a digital twin document. This way, algorithms can infer information about the twin and the real-world entity and new use-cases like the exchange of twins or the automatic aggregation of heterogeneous data are possible.

In [D5.4] the semantic twin ontology was already presented, and didn't require major overhauls for this final version. In the following we will summarise the basic concepts in the ontology, explain the changes since the initial version and describe the development procedures for future enhancements.

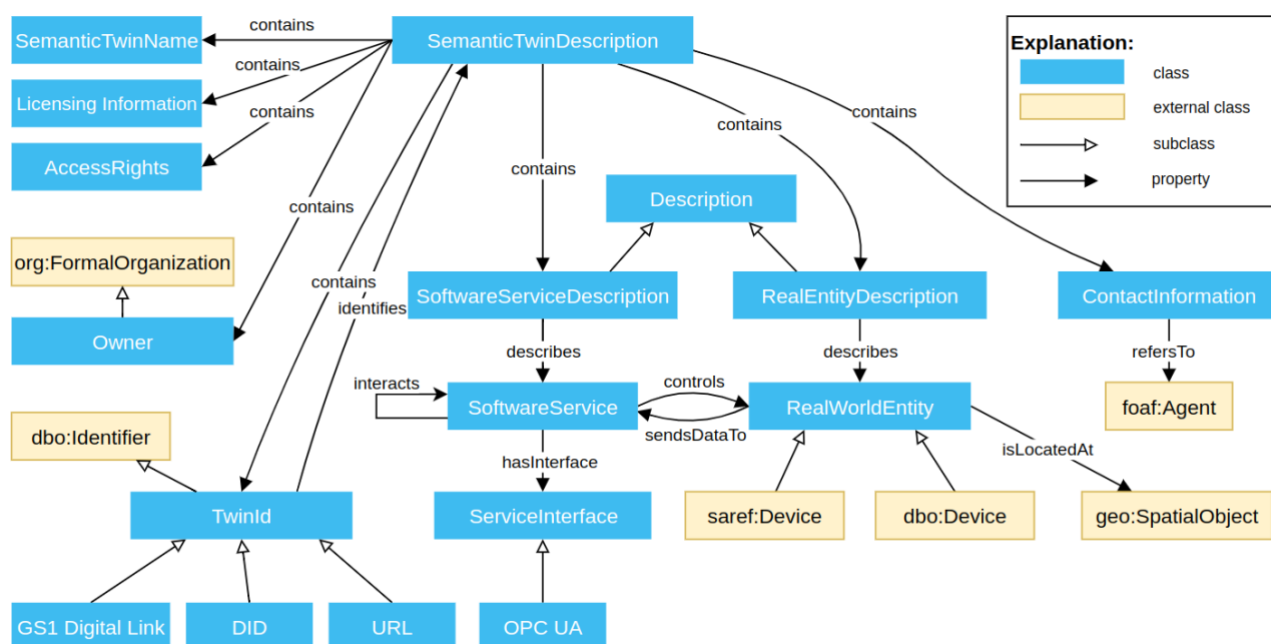


Figure 4.6 – Main Classes in the Semantic Twin Ontology.

Semantic Twins describe the essential information about a Digital Twin, namely the identity and owner of the Digital Twin, its real-world counterpart, access rights and terms of use, and relations to other Digital Twins. The main classes forming the Semantic Twin Ontology can be seen in Figure 4.6, where we find a central *SemanticTwinDescription* class containing relations to the relevant classes covering the aspects of the Semantic Twin. The class *SoftwareServiceDescription* describes the Digital Twin, whereas *RealEntityDescription* describes the physical counterpart. It can be observed that these have a described relation to a generic class, which can be set as a superclass to classes coming from other ontologies, which are focusing on their special domain. Thus, the Semantic Twin ontology can be used as a link between these ontologies. The different meta information of the Digital Twin do have their respective classes that reuse standard ontology classes where possible (e.g., the *friend-of-a-friend (FOAF)* ontology or the *DBpedia* ontology). Figure 4.7 uses a fictional example of a powertrain by the partner ABB to illustrate the use of the ontology.

Since the previous version, the most significant changes to the ontology were the usage of *owl:dataProperty*, which is meant to "connect individuals with literals" [OWL-W3]. These allow us to specify types of literal data, such as strings or integers, to be connected to individuals in the ontology. In the context of semantic twins, these are especially useful, as e.g., the name of a Twin or the human-readable description are such native data types. The *twinschema.org* ontology contains the following *dataProperties*: *DID*, *Random Number*, *TwinName*, *URL*, as well as *hasDescription*, which contains three subclasses: *hasCloudServiceDescription*, *hasRealEntityDescription*, and *hasSemanticTwinDescription*. This is separated, as the "Domain" (the classes that the property is assigned to) of these properties must differ. Otherwise, a reasoner would conclude that a *CloudService* and a *RealEntity* are the same thing, as both have a *hasDescription* property.

The ontology can be found at the project's GitLab repository⁹. We also use Gitlab Pages to provide an interactive ontology documentation browser, as is shown in Figure 4.7. This documentation is created using the "ontology-browser" plugin from Protégé¹⁰. This might be enhanced by using another tool for the html generation, but an extensive research on available alternatives could not bring up viable alternatives yet, as all results were either highly customised, undocumented or outdated.

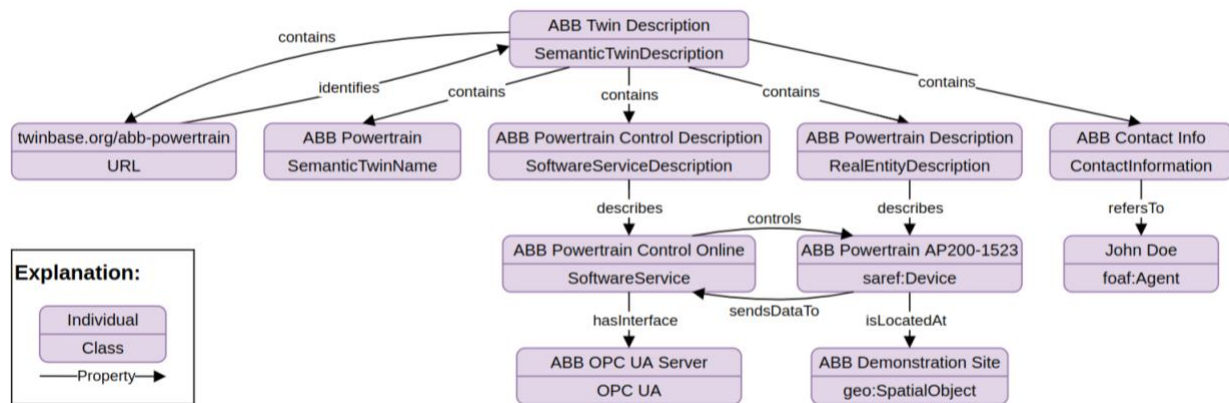


Figure 4.7 – Application of the ontology to a fictional powertrain example use case.

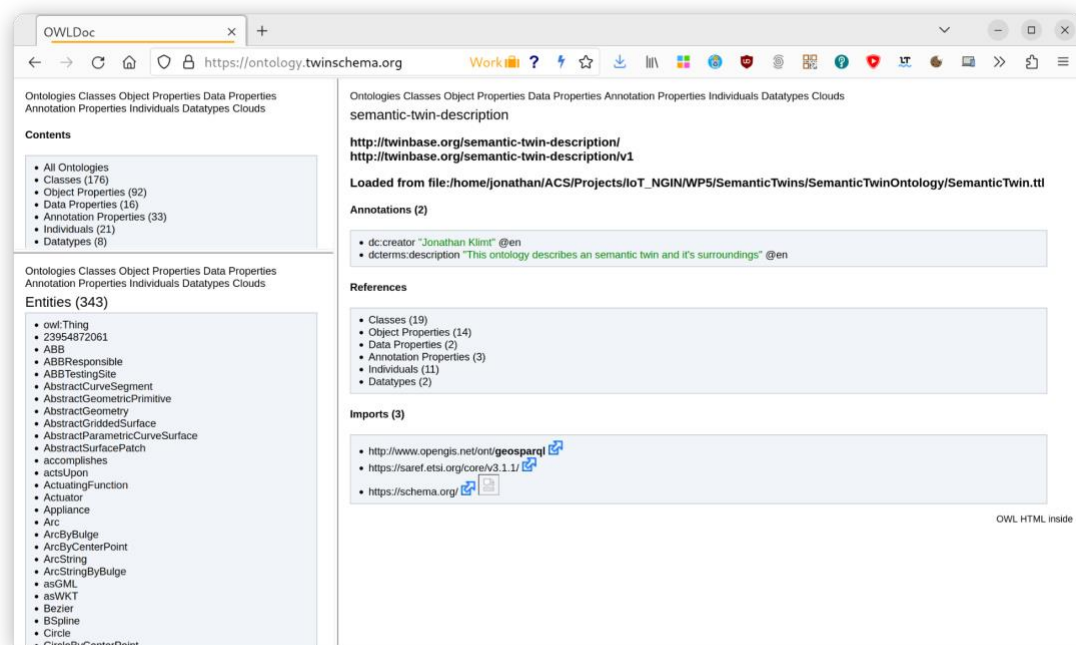


Figure 4.8 – The Ontology documentation browser on <https://ontology.twinschema.org>.

⁹<https://gitlab.com/h2020-iot-ngin/enhancing-iot-cybersecurity-and-data-privacy/semantic-twins/semantic-twin-ontology>

¹⁰ <https://github.com/co-ode/owl-plugins/ontology-browser>

The ontology for the powertrains in UC8 has been developed as described in previous sections. The main elements were taken into account in the solution so that suitability and usability of semantic twin approach can be tested.

However, in reality there are huge amounts of different setups in real applications: e.g. the number of main components like motors and drives can be different, depending on installations different sensors may be used, and there could be different kinds of users for powertrains. This means a lot of different needs and therefore it is not possible to include all needed ontologies for powertrain in this exercise. It is important to know what the possibilities are when using this kind of approach. At this point it is obvious that semantic twin approach is a good framework for UC#8.

Keeping the different needs in mind, the following aspect merit further consideration in the future:

- How to modify semantic twin document content easily as there are a lot of variations in powertrain setups? Manual work should be avoided and user-friendly interfaces should be available for operators.
- What is the automated procedure to add new data to twin documents?
- What are the required actions to add e.g. a new sensor for the system?

DRAFT - PENDING EC REVIEW

5 Decentralised Interledger solution

This section presents the Decentralised Interledger Bridge (DIB) solution. First, the section summarises the need for multi-ledger transactions and how they can be met with a suitable interledger solution, the IoT-NGIN requirements for the interledger, and the existing interledger approaches. Based on the requirements, the Flexible Interledger Bridge (FIB) [Wu2021] developed in the EU Horizon 2020 project SOFIE [SOF2021] was chosen as the basis for developing a decentralised solution, the *Decentralised Interledger Bridge (DIB)*. More details about available multi-ledger solutions and rationale for selecting the FIB as the basis of DIB can be found in IoT-NGIN deliverable D5.3 [D5.3]. The rest of the section details the DIB solution, both the original Ethereum DSM version and the more recent High-Throughput Fabric DSM version.

5.1 Motivation for Interledger

Interledger technologies enable transactions that span two or more distributed ledgers. This section summarises why a separate technical solution is required for linking the ledgers, what benefits this approach enables, and what requirements a good interledger solution has to meet to be able to address the needs of IoT-NGIN.

5.1.1 Need for multi-ledger transactions

Distributed Ledger Technologies (DLTs) have been developed for over a decade, and they have been widely adopted due to the immutability and transparency provided by the decentralised secure storage, the distributed trust ensured by sophisticated consensus algorithms, and the automatic execution within the system enabled by features such as smart contracts [Zha2019]. According to their individual design goals, different DLTs have a varying emphasis, including the accessibility of data on the ledger (i.e., who is allowed to read or write on the ledger), the consensus mechanism adopted to reach agreement on ledger status, and the range of supported functionalities.

As DLTs have been deployed to more application areas, it has become clear that no single DLT is suitable for all use cases. Sometimes even the requirements of a single complex use case can easily exceed the strengths and capabilities of any single DLT. In such situations, combining multiple DLTs with different strengths and features can be a beneficial approach as it enables new functionality [But2016]. For instance, it might help improve the data integrity by utilising a highly trustworthy public ledger, while reducing the cost and latency of a system by keeping most of the heavy-lifting business logic in private ledgers.

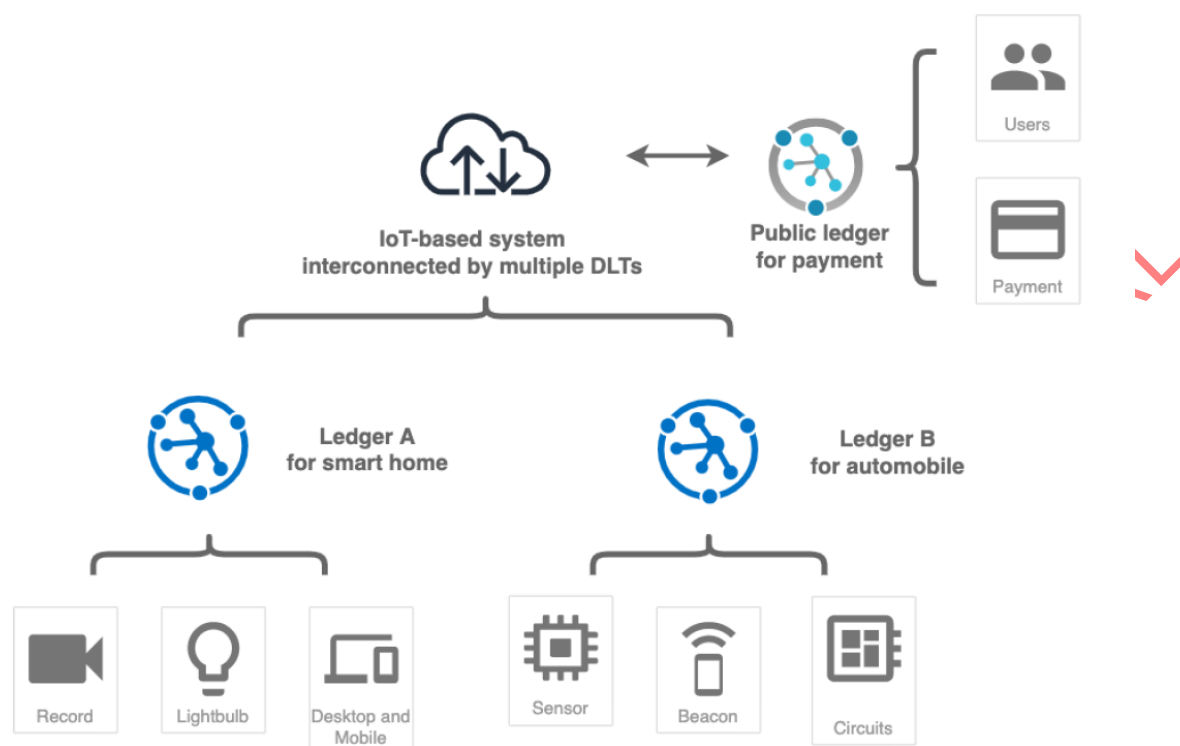


Figure 5.1 – An IoT-based system combining multiple DLTs.

A typical example are Internet of Things (IoT) systems, where an information sharing mechanism across multiple DLTs could help resolve the security, maintenance, and authentication issues in an automated manner [Has2019]. As illustrated in Figure 5.1, it is typical that IoT devices and services are connected to and backed by private distributed ledgers of individual vendors so that, e.g., the devices and equipment for a smart home interact with Ledger A, and the automobile sensors and circuits work together with ledger B. Then, a public ledger could be used for providing services for authentication and payment, and interlinking these three DLTs would enable a more complex (eco)system with additional functionality, e.g. payment services could be used with automobile ledgers at electricity charging stations.

5.1.2 Requirements of IoT-NGIN

The Interledger solution being developed (from here on: interledger) will be used in the IoT-NGIN project in several ways including Twin Smart Cities, Industry 4.0, and Smart Energy living labs, and also the Semantic Twins use case from WP5. Furthermore, the IoT-NGIN architecture is expected to introduce many other uses for the interledger beyond the IoT-NGIN project itself including the IoT-NGIN open calls.

More specifically, in the Smart Energy Grid Living Lab the energy marketplace data needs to be stored. In WP3, trusted AI is targeted for machine learning: Zero Knowledge Proofs (ZKPs) of training datasets and trained machine learning models together with its parameters can be automatically stored on DLTs in form of hash values and later utilised for verification by third parties to ensure they have not been tampered with, while no actual data is released

on the DLTs. Finally, Semantic Twins in WP5 utilise DLTs in a similar pattern, to ensure the integrity of twin documents.

All the above use cases require auditability for logged data, but storing everything in a highly trustworthy public ledger would result in high costs and expose all data to potentially prying eyes. The low throughput of public ledgers can also become a problem in some cases. Storing everything in a private ledger would protect privacy, provide better throughput, and slash costs, but would also lack the high level of trust. A solution is to store the data in the private ledger and then leverage an interledger to automatically store a hash of the data at suitable intervals to the public ledger. To improve privacy, this hash can also be salted by adding a random number to the calculation of the hash value to prevent guessing the data stored in the public ledger. This way, it is easy to verify whether the data in the private ledger has been tampered with while the overall costs are kept significantly lower as the usage of the expensive public ledger is reduced drastically.

Based on these different uses discussed above, 7 key requirements for the interledger solution can be identified as listed in Table 5.1 (table 2.1 in D5.3) and are detailed in the following text.

- REQ_IL_NF01: The interledger must be able to support the transfer of different types of data (so this excludes e.g. interledger solutions that focus exclusively on value transfers). Also, depending on the use, different types of DLTs may be utilised as part of the system, so the interledger solution has to be adaptable to different DLTs with relative ease.
- REQ_IL_NF02: The interledger must guarantee that the transactions across the ledgers are atomic, i.e. they happen completely on all the involved ledgers or not at all.
- REQ_IL_NF03: The interledger must provide transparency to the operations so that the correct operations of the interledger can be verified based on the data on the ledgers.
- REQ_IL_NF04: The interledger must operate so that non-repudiation for all parties of each individual transaction is guaranteed.
- REQ_IL_NF05: The interledger must be designed so that it can support a large number of transactions per second.
- REQ_IL_NF06: The interledger should minimise the overhead (cost, performance, storage etc.) for the application utilising the component for cross-ledger communication.

As detailed in D5.3, a centralised bridging solution, the Flexible Interledger Bridge (FIB) [Wu2021] developed in the EU Horizon 2020 project SOFIE, already meets all of the above requirements. However, as a centralised solution, it suffers from the known limitations of all centralised solution, namely higher trust requirement on the party running the bridge and lower resiliency. Therefore, the requirements list include one last requirement:

- REQ_IL_NF07: The interledger itself must support decentralisation, i.e. that the functionality is provided by a consortium of parties so that none of them can misbehave in any data transfer (e.g. change data payload, report invalid ledger transaction, or reject the transfer) without being detected by others. As a contrast, an interledger run by a single party has several limitations: the party has to be trusted by all users and it forms a single point of failure that can also pose problems for the

resiliency and performance of the solution; a decentralised interledger helps address these limitations.

Table 5.1 - Requirements for the interledger solution [D5.3].

ID	Requirement	Description
REQ_IL_NF01	Generality	Must support general-purpose data transfers and be easily adaptable to different types of distributed ledgers.
REQ_IL_NF02	Atomicity	Must guarantee atomicity of transactions across the ledgers.
REQ_IL_NF03	Transparency	Must be transparent enough that the correct operation of all transactions can be verified based on the data on the ledgers.
REQ_IL_NF04	Non-repudiation	Must support non-repudiation so that the participants to a transaction cannot later deny their actions.
REQ_IL_NF05	Scalability	Must support a large number of transactions per second.
REQ_IL_NF06	Efficiency	Should keep the application overhead low.
REQ_IL_NF07	Decentralisation	Must support decentralisation, where the interledger is run by a consortium of parties.

5.2 Detailed description of the developed solution

This section describes the Decentralised Interledger Bridge (DIB) in more detail. The DIB implementation has been published as open-source¹¹.

Compared with its single node predecessor, the decentralised architecture of DIB design provides the shared trust among a consortium of participants for interledger transactions, while improving the resiliency of the interledger data transfer via redundancy of nodes. To achieve a reasonable decentralised architecture for interledger, it is critical to make the following assumptions:

- Endpoints, which typically are smart contracts on DLTs, at both source and destination of a data transfer will implement the interfaces required by DIB.
- All interledger nodes in a consortium (regardless of the party controlling the node) have the same access to the endpoints, including both read and write operations.
- All nodes are equal in the sense that there is no special or admin node with superior functionality or access rights.

¹¹<https://gitlab.com/h2020-iot-ngin/enhancing-iot-cybersecurity-and-data-privacy/decentralised-interledger-bridge-dib>.

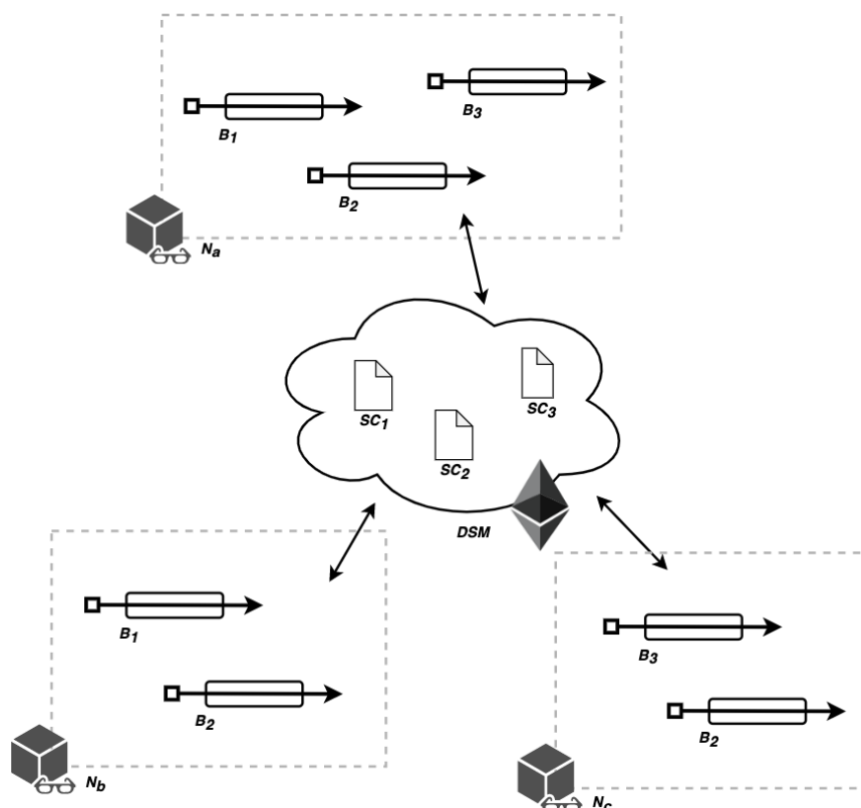


Figure 5.2 – DIB architecture consisting of nodes (N_x), bridge instances (B_x), and smart contracts (SC_x).

The high-level structure of the DIB design is illustrated in Figure 5.2. The architecture consists of identical *DIB nodes* implementing the bridge functionality and a *Decentralised State Management (DSM)* layer in the centre for synchronising the nodes. In the illustration, endpoints (typically smart contracts on a distributed ledger) of each connection are ignored for simplicity. The Connection Smart Contract (SC_x in the figure) on DSM manages *unidirectional* interledger transfers between certain endpoints.

In this decentralised architecture, the interledger nodes should always have access to the DSM layer that is shared among the consortium of partners. The current implementation supports use of Ethereum and Hyperledger Fabric for the DSM layer. Ethereum was chosen due to wide availability of tools and ease of deployment. However, Ethereum has not been designed for a high throughput, therefore Hyperledger Fabric should be used as a DSM layer in situations where the high performance is needed. DIB also supports a local state manager which resides in the node's memory for cases where the extra resilience is not necessary and a single-node setup is sufficient. The local state manager also has higher performance than DSM as it does not have to synchronise the activities with other nodes.

While all the nodes have access to the DSM layer, only a single node should perform a transaction to the endpoint ledgers. Here DIB supports a timeout mechanism to provide extra resiliency: if the node that is supposed to perform an endpoint transaction does not perform it within a certain amount of time, which is freely chosen by the deployer of the DSM, another node will take over this task.

There are two alternative versions of the DIB with different emphases. The original *Ethereum DSM based DIB* (described in D5.4 [D5.4]) focuses on active monitoring of the nodes to ensure

high trustworthiness and quicker recovery from unfinished transactions, but at the cost of lower throughput. The newer *High-throughput Fabric DSM based DIB* has the opposite priority by focusing on throughput at the cost of less active monitoring and slower recovery.

5.2.1 Data flow of the Decentralised Interledger using High-Throughput Hyperledger Fabric DSM

The downside of the active monitoring in the Ethereum based DIB is that it reduces the throughput of the bridge. To address this, an alternative DIB design based on Hyperledger Fabric has been developed to emphasise the throughput at the cost of monitoring and recovery speed. While the architecture of this Fabric DSM is different, the overall goal is the same as with the previous design: to increase the resiliency and trustworthiness of the bridge by supporting the collaboration of 1-N nodes that together provide the service, ensure that the other nodes have performed their work correctly, and (after the defined timeout) take over for failed nodes.

In the Fabric DSM implementation, the coordination between nodes and the tracking of the transfer process is minimised: one node is chosen to independently carry out the complete transaction and then write the outcome to the DSM. The other nodes then verify the transaction was carried out correctly and indicate their agreement with silence (i.e. they only write to the DSM if they *disagree* with the transaction). Alternatively, if the DSM write about the completed transaction fails to materialise within the (relatively long) timeout window, the failover node will start the recovery process and take over the transaction.

This design significantly reduces the time each node has to interact with the DSM and also reduces the load on the DSM, which in both designs will be a bottleneck after enough nodes have been added to the bridge. As a downside, this Fabric-based DSM does not support real-time monitoring of the transfer process, which results in much slower detection of anomalies. Further, the transfer timeout needs to be significantly longer than in the Ethereum based DIB as the whole transfer process including the client ledger processing all needs to be able to complete before the timeout is triggered, so recover from e.g. a node crash will take longer.

Furthermore, the recovery process was re-designed to support the new design where DSM doesn't have the intermediate states but the node taking over in a case of failure has to figure out the state of the transfer by querying initiator and responder ledgers. Also, the verification process was re-designed. Transfer is reported to the DSM layer only at the completion of the transfer to minimise the slow down due to the large number of DSM transactions. 4) Finally, the technology of the DSM layer was changed from Ethereum to Hyperledger Fabric for higher performance, and the different architecture of Hyperledger Fabric was utilised in the architecture of the DSM layer operations.



Coordination of the Transfer Process

In the Fabric-based DIB, each transfer is dedicated to one node, whereas in the previous design nodes compete to take the next action in each step of the transfer. The idea of the competition is to increase the trustworthiness of the bridge by distributing the responsibility to multiple nodes. However, it requires recording every step of the transfer to the DSM layer to coordinate the process between the nodes. This dependency to the DSM layer during the transfer process makes it a bottleneck of the overall throughput. Thus, in the new design the

dependency to the DSM layer was reduced drastically. The transfer is only recorded to the DSM layer for auditioning *after the transfer is finished*. Further, each transfer is dedicated to one node to avoid the need to communicate and synchronise between the nodes during the transfer.

This is achieved with a scheduling similar to round robin where nodes take turns to conduct transfers. As the new design follows the same principle as previous designs, clients are allowed to use arbitrary ids, even colliding ones, and thus the bridge has to generate a unique internal id for each transfer. The unique transfer id is generated with a deterministic algorithm based on the details of the source transaction, enabling each node to independently generate the same id. The transfer is assigned to a node by calculating the modulo of the transfer id and the node count, which results in the node id that the transfer will be assigned to. As transfers are assigned to nodes by deterministic algorithm, the ids are shuffled with a hash function to make it harder for the potentially malicious client to predict the node the transfer will be assigned to. The objective is to prevent a denial-of-service (DoS) attack where one node would be crashed by crafting transfer of which all would be assigned to that node only. The modulo based algorithm that assigns the transfers to nodes requires the number of nodes to be known and also the nodes need to have consecutive ids. The disappearance of a node will be taken care of by the recovery process, but adding new nodes requires updating the node count and ensuring the numbering remains consecutive.

DRAFT - PENDING EC

Transfer Process (Phases I-III)

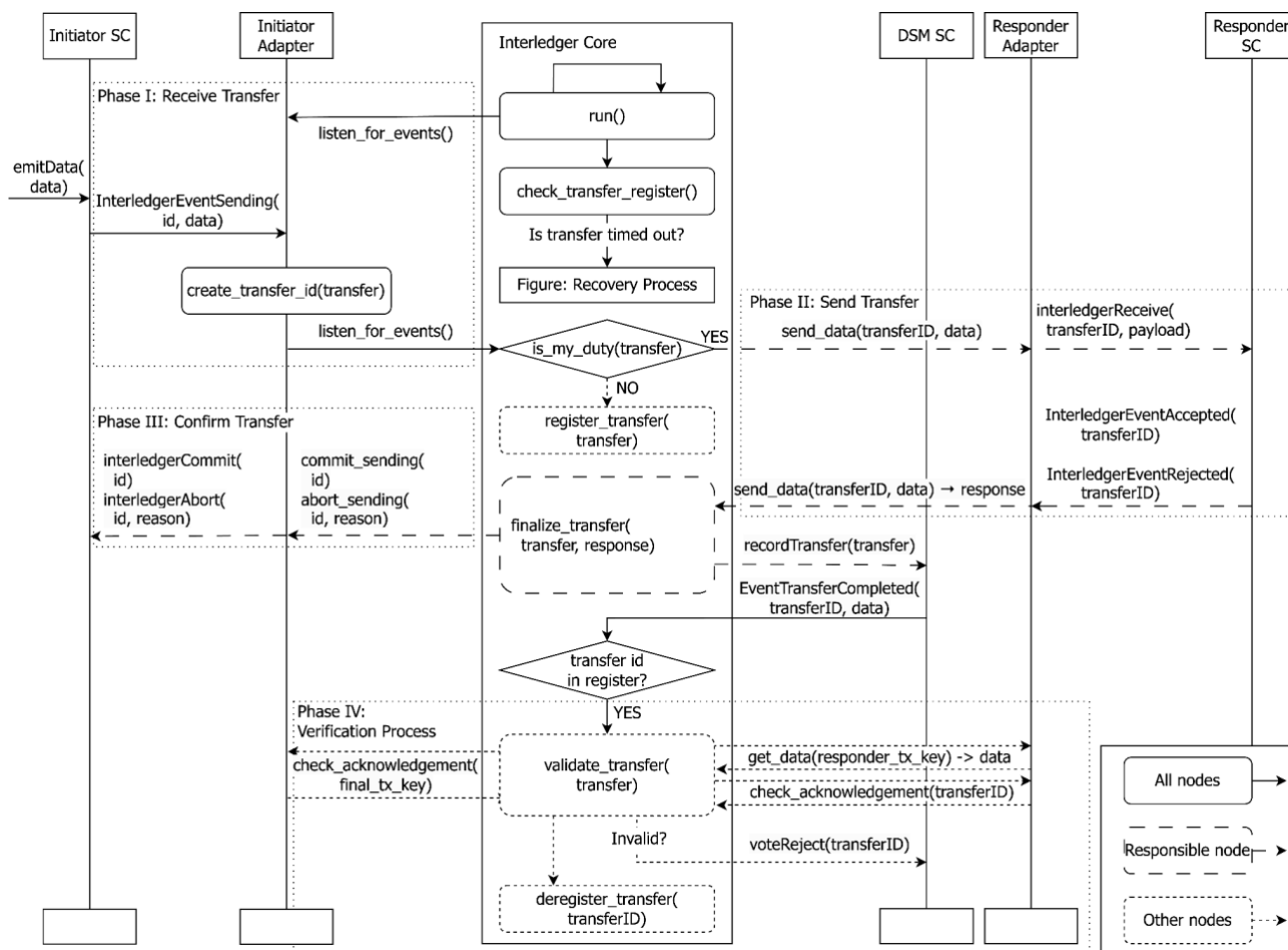


Figure 5.3 – Interledger transfer process using Hyperledger Fabric based DSM.

The transfer process consists of 3 phases as illustrated in Figure 5.3.

Phase I: Receive Transfer

1. All nodes will receive the *InterledgerSending(id, data)* event from the source ledger (Initiator).
2. All nodes calculate the unique transfer id with a deterministic algorithm based on the details of the source transaction, resulting in the same id in all the nodes.
3. The function *is_my_duty()* decides which node will start the transfer process, based on the id of the transfer and the id of the node.
4. Other nodes will internally register the transfer for monitoring the possible timeout in the process.
5. All nodes also store the identifying information of the originating event to the in-memory transfer object for later use.

Phase II: Send Transfer (node in charge)

1. The node in charge of the transfer calls *interledgerReceive(id, payload)* in the responder smart contract, where *id* is the unique id generated in the previous stage and data or payload is the content that is being transferred. (*id* is passed to the functions so that the event sent as an acknowledgement can be identified)
2. Responder smart contract will respond with an event of either *InterledgerEventAccepted(id)* or *InterledgerEventRejected(id)* depending on whether the recipient accepted or rejected the transfer. The response event can be connected to the right transfer with the *id*.
3. Identifying information of the data sending transaction is stored to the in-memory transfer object for later use.

Phase III: Confirm Transfer (node in charge)

1. The transfer will be acknowledged as committed or aborted to the initiator using *commit_sending(id)* or *abort_sending(id, reason)* functions in the initiator adapter and *interledgerCommit(id)* or *interledgerAbort(id, reason)* functions in the initiator smart contract respectively, based on the response received in the previous stage from the responder ledger. The *id* is the identifier of the data item in the originating ledger and it was stored in the node in the first stage, as mentioned.
2. Identifying information of the acknowledging transaction is stored to the in-memory transfer object for later use.
3. After the transfer process is completed, it will be recorded to the DSM to indicate whether it was successful (committed) or not (aborted), using *recordTransfer(transfer)* function, where the transfer parameter is basically the in-memory transfer object, which represents all the critical information of the whole transfer process. Calling *recordTransfer()* emits *EventTransferCompleted* event if the function is successful, and thus notifies other nodes about the completion of the transfer.

Recovery Process

Recovery process is designed for handling scenarios where the node in charge of conducting the transfer doesn't do its job as expected, for instance due to crashing. Similarly to the transfer process, the recovery process is also designed so that the nodes don't need to communicate with each other to coordinate the process, keeping the synchronisation and thus the utilisation of the DSM layer minimal. There is a pre-defined time window for each node in which transfer needs to be completed or it will be taken over by the next node.

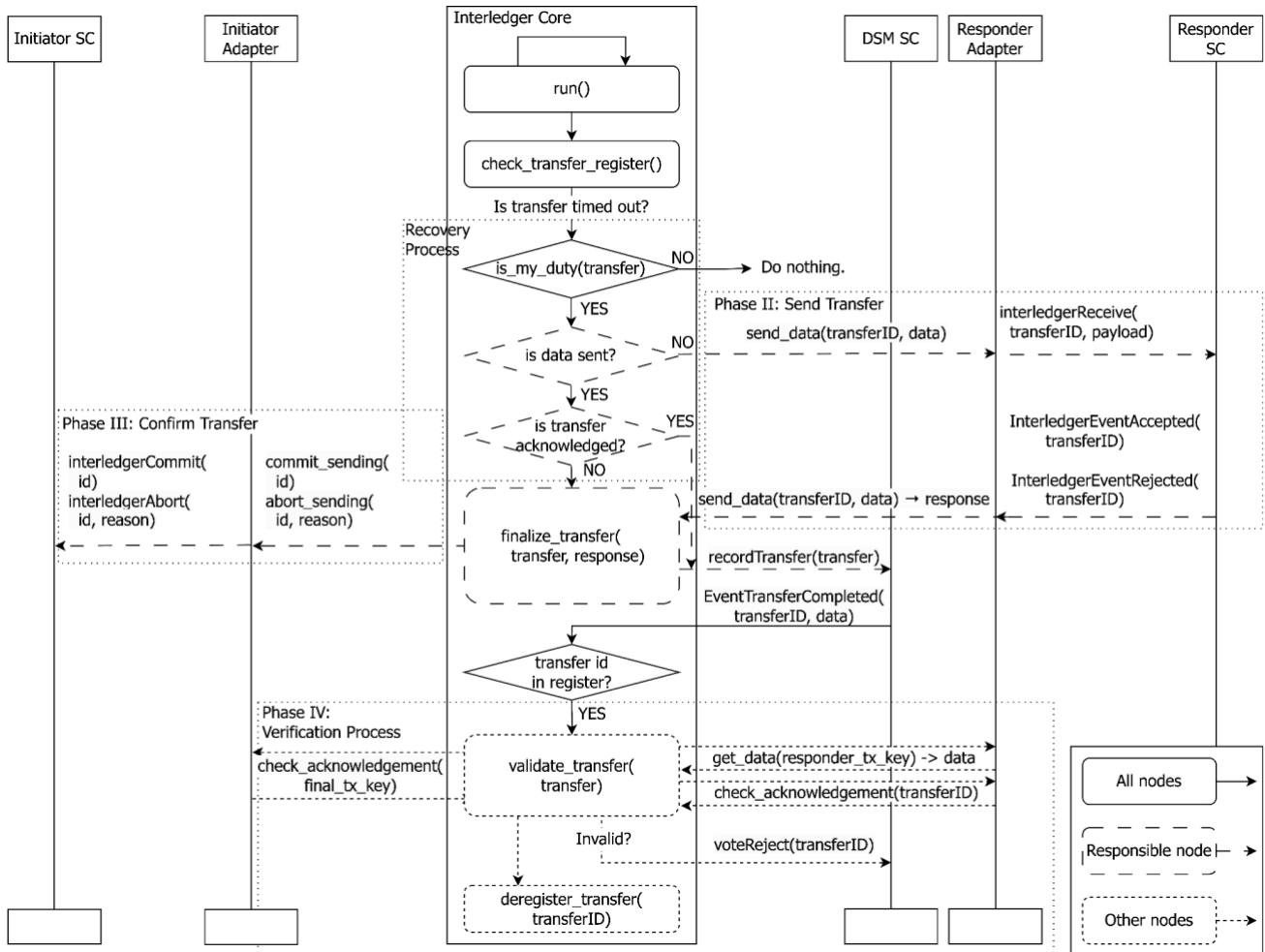


Figure 5.4 – Recovery process of Interledger transfer.

All nodes continuously scan all the transfers in their registries to detect transfers that have timed out. As described earlier, transfer is assigned to a node based on a unique internal transfer id and id of the node, but there is also a component that increments over time, more precisely over the timeout period, and thus the node selection is rolling. This mechanism guarantees that if a node becomes unavailable, responsibility for the transfer will eventually be assumed by another node without requiring explicit coordination between the nodes, and the transfer will be resumed.

When the timeout procedure activates, it starts by checking where the process was left unfinished and then continues from there as normally, as illustrated in Figure 5.4.

Verification Process (Phase IV)

To enable auditing, and thus, increase the trustworthiness of the DIB, all transfers are reported to the DSM after completion to generate an auditable log of the transfer process, including identifying information of each step of the transfer process. When a transfer is reported as completed to the DSM, the DSM also informs every node about the completed transfer that can be audited, and if flaws are detected, transfer can be flagged by voting for rejection.

Compared to the previous design of the DIB, endorsing is signalled by silence to reduce the load of the DSM which was the bottleneck of the system. As a result, there is only one voting round when the transfer is completed and reported, and in that voting round only rejection votes will be cast. This is possible due to a design where there is a time window for voting and if no rejection votes are cast during that time period, transfer is considered valid. This strategy adds a significant delay to the transfer process finalisation compared to the Ethereum-based design, but crucially it enables achieving a higher overall throughput.

Also, the verification process and the voting process specifically is optimised for the overall throughput by utilising Fabric's "parallel" architecture by registering votes with distinct composite keys using transfer's id together with node's id and thus write operations of casting votes won't conflict. The final number of votes can be counted by running a (range) query that retrieves all the votes with a key that starts with transfer id. In other words, votes are collected as distinct records that are aggregated on demand. No mutations are done, that would require synchronisation of the operations, and thus slow down the process.

The verification process has following steps:

1. All nodes receive a signal of a completed transfer, `EventTransferCompleted` event carrying all the details of the transfer. All the other nodes, except the one node in charge of the transfer, will check that all the steps of the transfer were conducted properly. The initiating transaction doesn't need validation as all the nodes receive the content with the originating event.
2. First, the nodes check that correct data was sent to the responder by comparing the response to the data payload in the originating event and the recorded data in the DSM.
3. Then, the nodes also check that the transfer was correctly acknowledged to the initiator by getting the acknowledgment event from the responder and checking that the corresponding function either `interledgerCommit` or `interledgerAbort` was called.
4. If there are any mismatches, nodes vote for rejection using `voteReject(transferID)` within a pre-configured time window.
5. After verification, the transfer is cleared from the register.

5.2.2 Security properties of decentralised Interledger

The DIB provides decentralisation with the following benefits:

1. *Resiliency and availability.* If one DIB node is not available to participate for any reason (node is down, lack of network connectivity, etc.), the interledger transactions will be successfully completed by other DIB nodes, as long as there is a sufficient number of nodes available. Even if one DIB node has already indicated its willingness to perform the transaction and then it is not able to do it, another node will take its place after the timeout.
2. *Auditability.* The DIB design allows multiple nodes (and parties) to join the DSM layer, which keeps track of interledger transactions. Therefore, all parties are able to verify that the transactions have been performed correctly.

However, the DIB cannot prevent malicious node behaviour. Any node that has access to the source and destination ledgers can perform malicious transactions directly with these ledgers, bypassing the DIB altogether. For example, the malicious node can signal to the

source ledger that the transaction has been accepted/rejected immediately, or perform the *interledgerReceive()* transaction on the destination ledger with incorrect data or without the corresponding trigger from the source side. However, in these cases DIB still provides auditability, if all the nodes that have access to the source and destination ledgers participate in the DSM, then the malicious node can be identified by comparing transactions on the source, destination, and DSM ledgers. However, the current implementations do not try to monitor e.g. for *interledgerReceive()* transactions without a trigger on the source side, but such a functionality could be implemented relatively straightforwardly as no bridge node is supposed to interact with that function without a trigger.

By default a majority of nodes is sufficient to endorse/reject transactions, e.g. if there are 9 nodes in the DSM then endorsement from 5 of them is enough. This parameter can be freely chosen during the deployment of DSM smart contract, however changing it drastically may worsen the resiliency or security properties of DIB. E.g., if it is required that 90% of nodes endorse DSM transactions, then just having 11% of nodes offline or acting maliciously would stall the DIB process since there will not be enough nodes to endorse them. In situations where malicious behaviour of DIB nodes is not expected and the availability of individual nodes is poor, the sufficient number of endorsements can be significantly below 50% of the nodes.

5.3 Analysis of DIB features and performance

The DIB component satisfies all the requirements presented in Table 5.1:

- DIB supports transfer of any kind of data, instead of just monetary value. (REQ_IL_NF01)
- DIB provides atomic transactions, the transaction is confirmed/aborted on the Initiator ledger depending on the result of the Responder transaction. (REQ_IL_NF02)
- DIB provides transparency and non-repudiation since all of its actions are recorded to the ledgers. (REQ_IL_NF03 and REQ_IL_NF04)
- DIB component itself does not produce a high overhead and supports a large number of transactions. Performance and throughput of ledgers themselves is often the limiting factor. (REQ_IL_NF05)
- Ledger interfaces provided by the DIB component are simple and do not incur significant additional cost for the application smart contracts, running the component does not incur significant CPU overhead. (REQ_IL_NF06)
- DIB supports decentralisation as described in this section. (REQ_IL_NF07)

5.3.1 Initial DIB performance results

This section presents some initial DIB performance results, the final performance analysis will be presented in IoT-NGIN deliverable D6.3. In the DIB deployment there are several potential performance bottlenecks:

- Initiator and Responder ledgers, in many use cases one of these is a public ledger with very limited throughput
- DSM ledger of DIB
- The DIB component that is written in Python

During testing Ethereum ledgers utilising Geth¹² software were used as the Initiator and Responder ledgers. The testing revealed that the performance of these ledgers is around 20 transactions per second, which is quite low but understandable since Ethereum has not been designed for high throughput.

The following Table 5.2 presents intermediate test results of Gametoken [Gam2022] transactions in the following situations:

- Transactions performed directly, without DIB component, using a single test script
- Using DIB with local state manager
- Using 1-node DIB with Ethereum DSM (Geth)
- Using 3-node DIB with Ethereum DSM (Geth), with all nodes running on the same server
- Using 1-node DIB with Hyperledger Fabric DSM
- Using 3-node DIB with Hyperledger Fabric DSM, with all nodes running on the same server

Table 5.2 - Initial performance results for DIB: Throughput and standard deviation (in parenthesis).

Scenario	Throughput (standard deviation)
<i>Direct process (no DIB used)</i>	11.64 (0.1302)
<i>1-node DIB with local state manager</i>	11.22 (0.2748)
<i>1-node DIB with Ethereum DSM</i>	2.84 (0.0949)
<i>3-node DIB with Ethereum DSM</i>	4.52 (0.1450)
<i>1-node DIB with Hyperledger Fabric DSM</i>	11.32 (0.1936)
<i>3-node DIB with Hyperledger Fabric DSM</i>	21.80 (0.6589)

Results indicate that in addition to the ledgers, also the DIB component can be a bottleneck, and therefore running multiple nodes in parallel increases the performance of both the Ethereum and Fabric based DIB, since only one node handles a single transaction. Overall, Hyperledger Fabric based DIB that has been optimised for high performance achieves good results without significant performance penalty compared to direct process and with 3-node setup the performance actually increases, indicating that the DIB component itself is a bottleneck. The performance of the DIB component could likely be increased by e.g. adding internal parallelism, but such optimisations were not the goal of this proof-of-concept implementation.

¹² <https://geth.ethereum.org/>

The next section discusses the differences between using the Ethereum and Hyperledger Fabric ledgers for the DSM layer in more detail.

5.3.2 Ethereum vs. Hyperledger Fabric as the DSM layer

There are several fundamental differences between the Ethereum and Hyperledger Fabric. In the Ethereum ledger all transactions in the block are ordered and executed sequentially (the order is decided by the miner), therefore the different transactions within the same block may safely modify the same data structures, and the Ethereum DSM implementation takes advantage of it.

In contrast, in Hyperledger Fabric transactions are executed concurrently using the execute-order-validate (EOV) model to achieve higher throughput. As Hyperledger Fabric doesn't force sequential execution, multiple transactions targeting the same key, where at least one of the transactions is writing, may cause a conflict. More specifically, a conflict arises, if during the time when the transaction is simulated on the peer (i.e. read-set is created) and it's ready to be committed, another transaction changes the value of the same key. Thus, the read-set version will no longer match the version in the orderer, and concurrent transactions will fail. This is addressed in the Fabric based DSM by designing each write to use a unique key and values are aggregated afterwards if needed.

Because of these differences, it was decided to optimise the Fabric DSM implementation for the maximal performance by reducing the information being written to the DSM. This makes detection of errors and misbehavior more difficult and slower, but still possible since all the necessary evidence exists in the source and destination ledgers.

As can be seen from the performance results, the DSM implementation based on Hyperledger Fabric is much faster, achieving up to 5 times higher throughput compared to Ethereum in these small deployments.

5.4 Practical use cases of DIB

DIB can be used in any application requiring transfer of data between ledgers. Example use cases include hash transfers to improve accountability and resilience, using ledgers to define and enforce access control policies, and using and trading virtual assets.

5.4.1 Transferring hashes to more secure ledger

In practical applications most of the ledgers used will be private (e.g. Hyperledger Fabric, Quorum, private Ethereum, etc.) since they offer vastly superior privacy, latency, throughput, and cost compared to public ledgers. However, immutability of the transactions on the private ledger is naturally not as strong as with a public ledger which is run by millions of nodes.

By taking a hash of multiple private ledger transactions (e.g. using a Merkle tree) and storing it to the public ledger (such as public Ethereum) would provide strong immutability guarantees (after the hash has been stored to the public ledger, it would not be possible to modify related private ledger transactions without breaking the hash) while keeping the costs low. This operation can be easily automated by DIB: periodically a smart contract on the private ledger is called by the application, and it calculates the hash, which is then sent

to the public ledger using the DIB process. The frequency of the hash transfer is application dependent: higher frequency would decrease the window of vulnerability of transaction modifications on the private ledger, but would also increase the costs.

In addition to the public ledger, the destination ledger for storing hashes can also be a private ledger run by a larger number of organisations, therefore offering better security and immutability properties. Also, this hierarchy can contain more than 2 layers, so, for instance, data from the private ledger can first be hashed to an intermediate ledger run by a larger consortium, which then hashes to a public ledger.

5.4.2 Using ledgers for Access Control

DIB together with multiple ledgers can be used for purchasing access control tokens in a secure and transparent manner [Sir2020], in this case one ledger is used for payments (e.g. a public Ethereum) while another one is used to provide access control tokens (e.g. a private Ethereum or Hyperledger fabric). The approach guarantees that only those who pay for the access will receive the corresponding access control token.

5.4.3 Trading of Virtual Assets

Suppose a situation where the virtual asset (e.g. an item in an online game) is backed by the ledger (to enforce uniqueness of the item and prevent creation of unlimited copies of it). This ledger would be a private ledger (Hyperledger Fabric, Quorum, private Ethereum, etc.) run by a gaming company or consortium of gaming companies.

In order to facilitate trading of the assets, a widely used public ledger (such as public Ethereum) is needed. The role of the DIB is to make sure that the asset is active only in one ledger simultaneously: either it's active in the gaming ledger and can be used for the gameplay and not traded, or it's active in the trading ledger in which case it can be traded but can't be used for the gameplay [SOF2020].

6 Self-Sovereign Identity Technologies

The use of SSI technologies for the triplet's identity and trustworthiness has already been discussed in Section 2. This section, therefore, provides details of the two other uses for the SSI technologies explored in IoT-NGIN, i.e. Verifiable Credential based decentralised on-device access control with constrained IoT Devices and QR code and GSI Digital Link based discovery mechanisms for the Triplet.

6.1 Verifiable Credential-based Access Control on Constrained IoT Devices

Verifiable Credentials allow flexible and privacy-preserving access control solutions. E.g., suppose there is a factory that has outsourced the maintenance to a separate company. The technician working for the maintenance company needs to receive temporary access to factory premises and to certain machines there, but the factory does not need to learn about the technician's real identity or whether the technician is the same as the one who visited the factory previously.

This subsection describes a verifiable credential-based access control solution that can be used directly on the constrained devices, i.e. the constrained device such as ESP32 microcontroller verifies the credentials and enforces the access control policies. The solution is also available as open source ¹³.

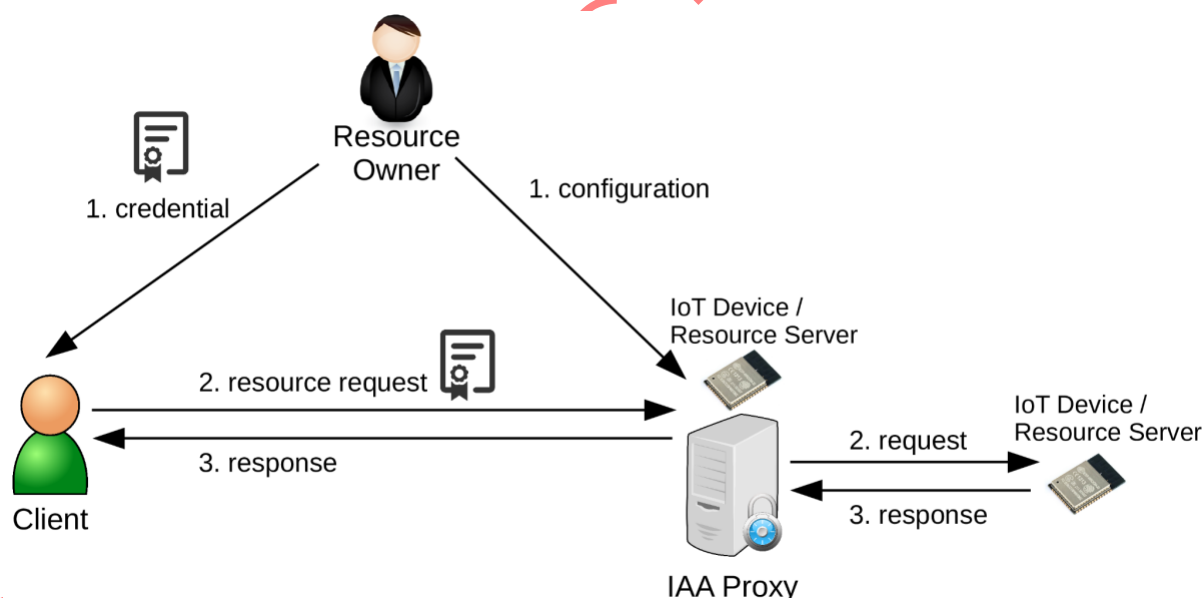


Figure 6.1 – Overview of the SSI Access Control component.

Figure 6.1 provides an overview of how the SSI component can be used to grant and verify access to the Resource Server, which can be for example an IoT device. The Resource Owner

¹³<https://gitlab.com/h2020-iot-ngin/enhancing-iot-cybersecurity-and-data-privacy/privacy-preserving-self-sovereign-identities>

and Client are identified using Decentralised Identifiers (DIDs). In the first step, the Owner configures the *Identity, Authentication and Authorisation* (IAA) proxy and grants a Verifiable Credential (VC) to the Client, which denotes that the Client has a right to access some Resource. In the second step the Client uses this credential to contact the IAA proxy or the actual IoT device, which will then verify the credential and grant a read or write access to the resource as shown in the step 3. In a case of the IAA proxy, it will forward the request to the actual Resource Server, which does not need to understand SSI technologies or even handle the cryptographic operations.

In more detail, the credential is encoded as a standard JSON Web Token (JWT) and in order to prevent replay attacks, the client also constructs a Demonstrating of Proof-of-Possession (DPoP) proof when accessing the resource. Both the credential and the DPoP proof will be verified by the IAA proxy or the actual device.

The SSI component provides the following functionality:

- Tools for identity and key management, including the creation of credentials encoded as JWTs and DPoP proofs. For DID methods, did:self and did:key are supported and the Ed25519 EdDSA signature scheme is supported for cryptographic signatures.
- IAA proxy and simple resource server based on existing py-verifier work¹⁴.
- Verifier for ESP32-based embedded devices, which allows full verification of access control credentials to be performed on an embedded device.

The performance on the constrained device is good, the full JWT + DPoP verification consisting of two signature verifications takes just 160ms on the low cost ESP32 device. Therefore, the whole process of accessing the protected resource takes well below one second, which is a sufficient performance from the user experience point of view [Fot2022].

6.2 Triplet discovery using QR codes and GS1 Digital Links

A GS1 Digital Link¹⁵ converts a barcode, either one or bi-dimensional, into a web address that contains the information on a product the barcode refers to. GS1 digital links are used to discover the locations of the Digital and Semantic Twin of an entity Triplet.

The discovery protocol begins with a user in front of a barcode, e.g. a QR code, attached to a real-world entity, such as an IoT device, and is shown in Figure 6.2.

¹⁴ <https://github.com/mmlab-aueb/py-verifier>

¹⁵ <https://github.com/gs1/GS1-DigitalLink-Resolver-CE>

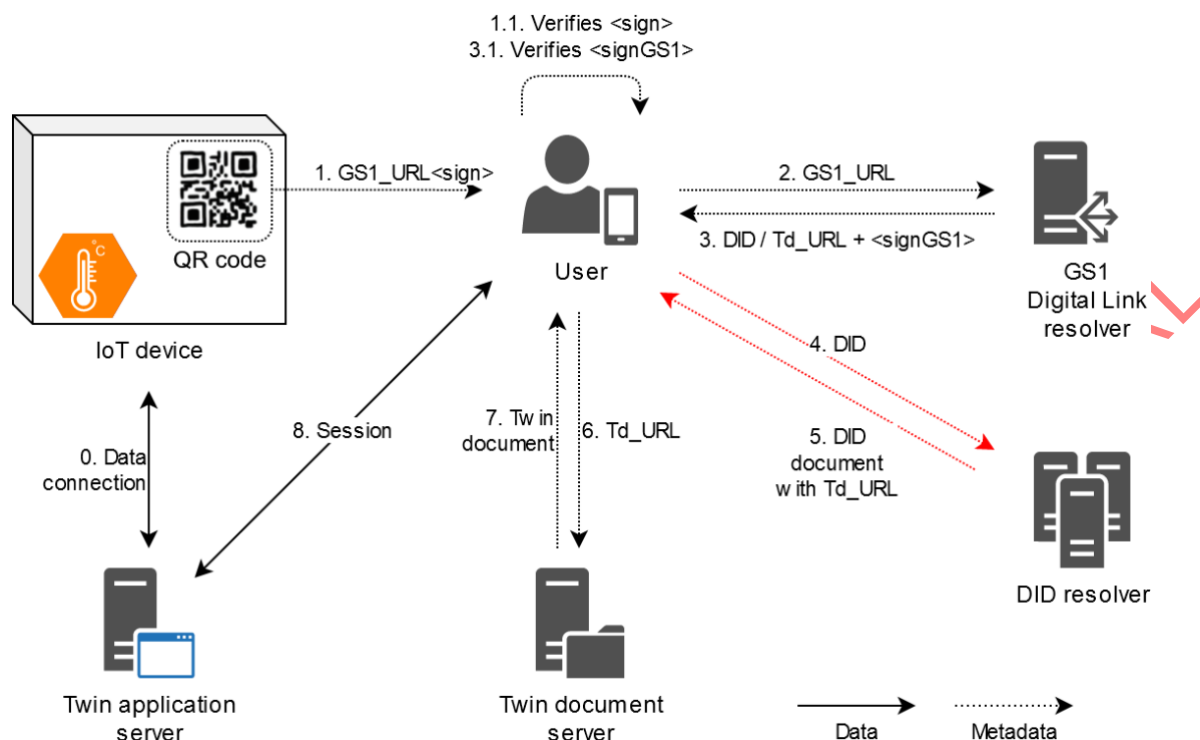


Figure 6.2 – The triplet discovery protocol.

The QR code encodes the URL and the Global Trade Item Number (GTIN) of the device to the GS1 Digital Link Resolver server¹⁶. The User scans the QR code with a smartphone using a dedicated app that queries the GS1 Digital Link Resolver Server to get either the locations or the DIDs of the Digital and Semantic Twins. This differentiation depends on the DID method used by the entity triplet:

- If the DID method is a ledger-based one that allows adding information to a DID into the ledger, such as *did:ethr*, the GS1 Digital Link Resolver server returns the DIDs of the Digital and Semantic Twins whose resolution, shown in red arrows in Figure 6.2, gives the User their DID documents containing the location parameters;
- Otherwise, if it is not possible to add data to DIDs, such as in *did:key* DID method, the Resolver server returns the User the location of the Digital and Semantic Twins.

In either case, the user accesses the digital or the Semantic Twin. The figure shows the user accessing the Semantic Twin and getting the Twin Document that allows them to open a session with a Twin Application Server and perform operations (depending on their level of privilege).

To guarantee the QR code the User is scanning is the original one, and it has not been switched with a malicious one, the QR code could embed the digital signature of the organisation that issued it¹⁷. This is feasible since a QR code can encode up to 3 KB of data. Before accessing the URL encoded in the QR code, the user's app verifies the signature with

¹⁶ <https://gs1resolver.iot-ngin.eu/gtin:123456>

¹⁷ <https://gs1resolver.iot-ngin.eu/gtin:123456&<digital signature>>

the organisation's public key (step 1.1 in the figure). Similarly, the data returned by the GS1 Digital Link Resolver server is digitally signed and verified by the User (step 3.1 in the figure).

The code of the GS1 Digital Link Resolver server can be found at Gitlab¹⁸.

DRAFT - PENDING EC APPROVAL

¹⁸<https://gitlab.com/h2020-iot-ngin/enhancing-iot-cybersecurity-and-data-privacy/ar-discovery>

7 Integrating the solutions

The solutions 5-8 can be used together to enable data sovereignty by making IoT data and services accessible in a trusted, auditable, and controlled way. Section 7.1 describes a demo for the configuration of IoT devices to showcase how the integration of the solutions works. Section 7.2 presents the Living Lab use cases that adopt, implement, and validate such integrations for all 8 solutions.

7.1 IoT devices configuration demo

This demo being prepared in WP7 for the Smart City LL integrates solutions 5-8 to demonstrate how to easily discover, protect, and configure the IoT Triplet while protecting the privacy of the individual users and providing good user experience through low-latency validation. The key actors of the demo use case are illustrated in Figure 7.1.

The *Traffic Department* of a City buys IoT Devices from a *Manufacturer* and wants to install them to a Smart City project. The Traffic Department initialises a device and its Digital Twin and Semantic Twin with the basic information required to delegate the setup to an external *Installer Company*. Moreover, the Traffic Department creates a QR code for each device, embedding a GTIN number that, once resolved by a GS1 Digital Link Resolver server, provides the locations to access the device's Digital Twin and Semantic Twin.

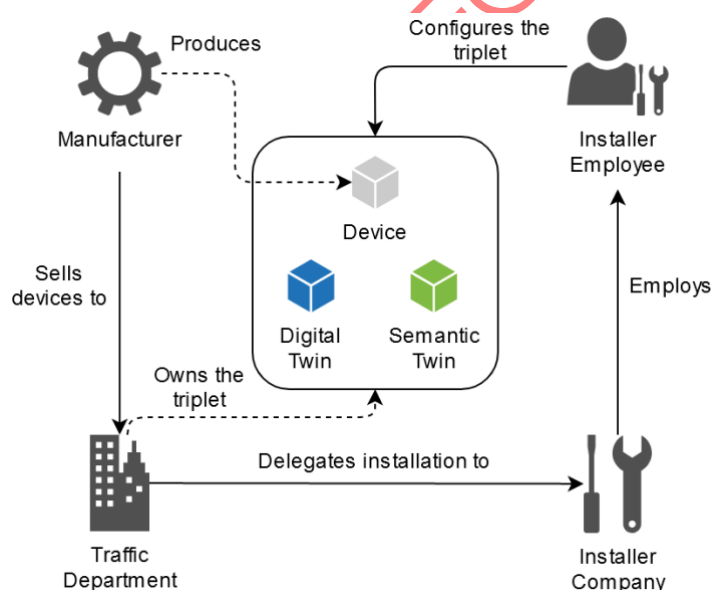


Figure 7.1 – Illustration of the demo.

The Installer Company employs one or more *Installers Employees* to go around the city and install the devices (and possibly maintain them afterwards). To finalise the installation of a device, an Installer Employee accesses the Digital Twin and the Semantic Twin of that device. To access them, they require a credential that can be obtained from the Installer Company Authorization Server. The Employee scans the QR code on the device with a mobile phone application. Once scanned, the QR code redirects the Employee to the Semantic Twin. At access request, the server hosting the Semantic Twin, e.g. the Twin document server shown in Figure 5.2, begins an access control protocol to know the privileges of the person who is requesting the access. With the QR code being accessible to anyone, any citizen of the City

could potentially get access to the Semantic Twin to view information about the Device. This could be a wanted feature of the Smart City project.

With this demo, we aim to address the following problems:

- Discovering the Twins related to an IoT Device;
- Enabling secure access to the triplet;
- Trusting the data received by the triplet;
- Protecting installer's privacy;
- Detecting malicious activities on the triplet.

7.1.1 Demo Description

The system resulting from the integration of the solutions described in this document is structured as follows.

The Traffic Department, who owns the entity triplets, is responsible for setting up the entries for each triplet in the GS1 Digital Link Resolver server and printing the correspondent QR codes. Moreover, the Traffic Department issues a VC_{Dept} to the Installer Company to configure the triplets: this VC has a "delegate" option so that the Installer Company can delegate the installation rights to its Employees.

The Installer Company sets up an Authorization Server that, being delegated by the Traffic Department, issues a VC_{Config} , alongside VC_{Dept} , to the Employees to configure the triplets.

Following the SSI approach, any actor issues or receives VCs from or to their DIDs. Figure 7.2 shows the DIDs paired to each actor in this demo. Actors such as the Traffic Department, the GS1 Digital Link Resolver server, and the Installer Company Authorization server may need to attach additional information to their DIDs. Thus they could use a ledger-based DID method. Instead, others may only need DIDs as pseudonyms, therefore a non-ledger-based DID method would be suitable.

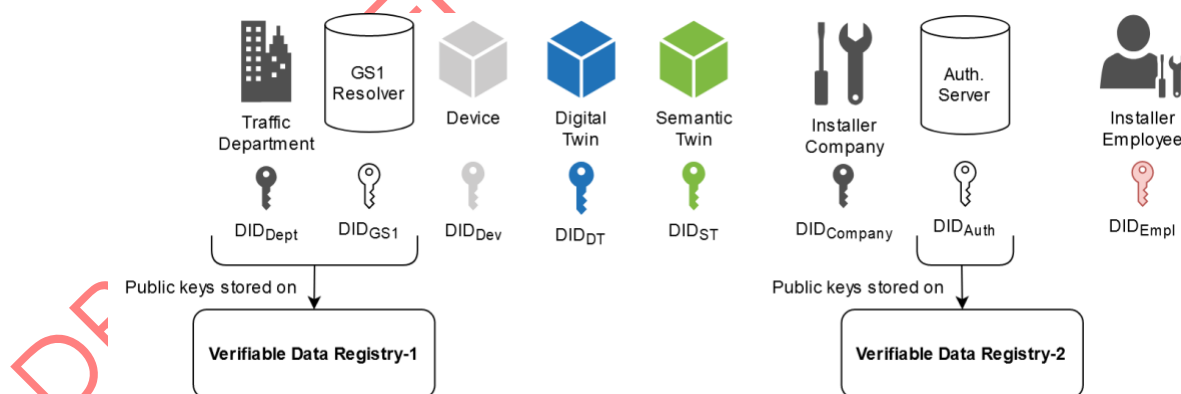


Figure 7.2 – Illustration of the DIDs used by the actors.

Before configuring the triplets, the Installer Employee generates their DID_{Empl} and requests the Authorization Server a VC_{Config} to be issued, alongside VC_{Dept} , to the newly generated DID. Examples of attributes, or claims, of VC_{Config} are the type of Devices the Employee will configure, their location, and the duration of the validity of the credential (e.g., 24 hours). The Installer Company and the Employee need to agree on a common secret parameter or

a similar solution to ensure only an Employee of the Installer is able to request such credentials.

When the Employee reaches a Device, scanning the QR code triggers the discovery protocol described in Section 6.2.

When the Employee locates the server hosting the Semantic Twin, they can access it following the access control protocol described in Section 6.1. In particular, the Employee signs a DPoP with their DID_{Empl} and sends it alongside the credentials VC_{Dept} and VC_{Config} to an IAA proxy to the Semantic Twin for access control. The IAA proxy checks:

- The validity period of VC_{Config} ;
- The attributes in VC_{Config} match its attributes (and the Employee is not accessing to the wrong device);
- The signature in VC_{Dept} is verified by DID_{Dept} ;
- The signature in VC_{Config} is verified by DID_{Auth} ;
- Ensure DID_{Auth} is delegated by VC_{Dept} ;
- The signature in DPoP is verified by DID_{Empl} .

If all checks are successful, the proxy allows access to the Semantic Twin to retrieve the Twin Document. To make the Twin Document data more trustworthy, the Semantic Twin can sign it with its DID_{ST} . Moreover, the security, integrity, and accessibility of the Twin document is helped by integrating DLTs and the Interledger component, whose functionality is described in Section 4. The protocol is shown in Figure 7.3.

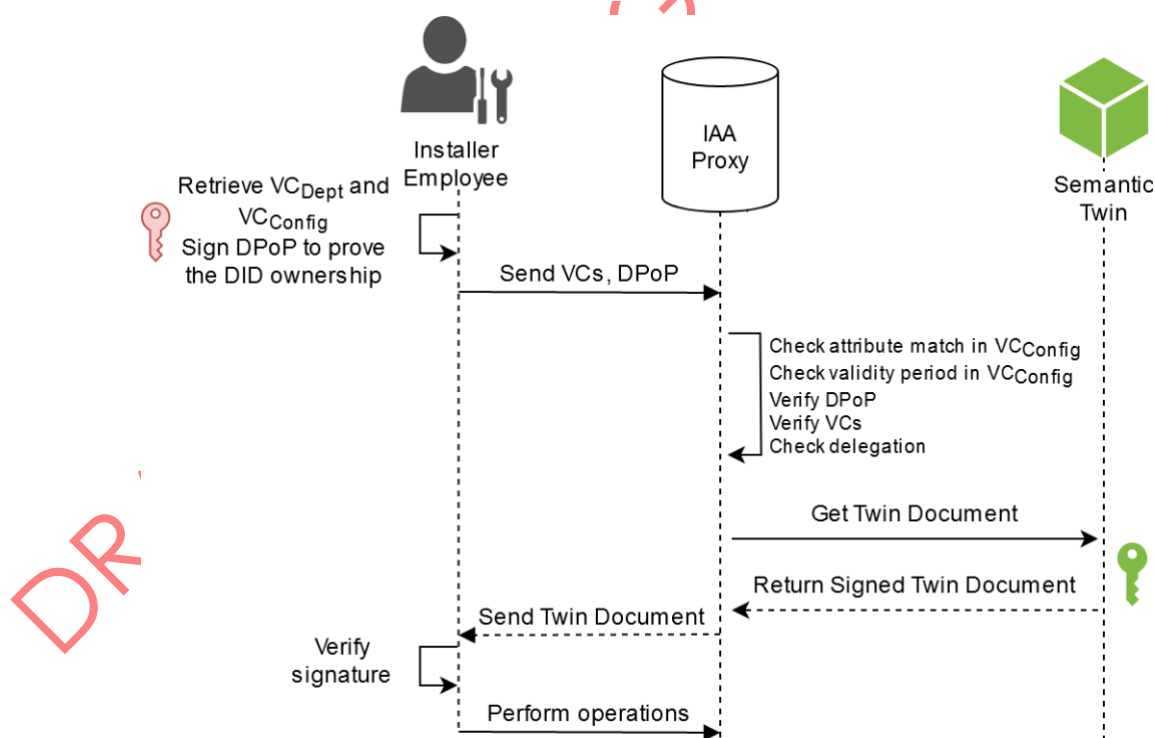


Figure 7.3 – The access control protocol to the Semantic Twin.

The problems mentioned in this section are addressed as follows:

- Discovering the Twins related to an IoT Device: this is solved by the GS1 Digital Link Resolver server;
- Enabling secure access at the triplet: this is solved by the VCs issued by the actors the access control protocol executed by the IAA proxy or on-device validation;
- Trustworthiness of the data received by the triplet: this is solved by applying digital signatures to the QR code, to the response data returned by the GS1 Digital Link Resolver server, and to the data returned by the Semantic Twin;
- Protecting installer's privacy: this is solved by hiding the identity of the Installer Employee during access time to the Semantic Twin behind an ephemeral DID;
- Accountability for malicious activities on the triplet: if a malicious behaviour is detected on a Semantic Twin, the Installer Company can link the DID used to access the Semantic Twin to the identity of the Employee who used that DID to request the VC used to access to the Semantic Twin, and take action.

DRAFT - PENDING EC APPK

8 Verification and Validation

Verification results of the solutions 1-8 will be reported in IoT-NGIN deliverable D6.3 - Interoperable IoT-NGIN meta-architecture & laboratory evaluation, which is due in June 2023.

The components will also be validated in the use cases of four IoT-NGIN living labs as described in Table 8.1. These validation results will be reported in the IoT-NGIN deliverable D7.4 - IoT-NGIN Living Labs use cases Assessment and Replication guidelines, which is due in September 2023.

Table 8.1 – Use of components in IoT-NGIN living labs.

IoT-NGIN technology	Smart Cities			Smart Agriculture		Industry 4.0			Smart Energy	
	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10
Generative Adversarial Networks (GAN) based dataset generation				X					X	X
Malicious Attack Detection (MAD)				X					X	X
IoT Vulnerability Crawler	X	X		X					X	X
Moving Target Defences (MTD) Network of Honeypots				X						
Decentralised Interledger Bridge	X	X						X		X
Privacy preserving Self-Sovereign Identities (SSIs)	X	X	X						X	X
Semantic Twins	X	X	X					X	X	

9 Conclusions

This document discusses the technical solutions from the IoT-NGIN WP5. These 4 to cybersecurity solutions for protecting IoT systems performing Federated Learning and 4 solutions for data sovereignty and privacy problems in the domain of IoT systems.

Based upon various needs in use cases within the IoT-NGIN project, technical solutions for each area were planned and successfully developed. The solutions are:

1. a GAN-based dataset generator for the creation of poisoned datasets that assist addressing attacks against IoT and Federated Learning systems
2. a Malicious Attack Detector (MAD) that facilitates the detection of cyberthreats and attacks against an IoT system
3. An IoT Vulnerability Crawler (IVC) that monitors IoT nodes and detects vulnerabilities
4. a Moving Target Defense (MTD) Honeypot Framework that deploys the honeypots dynamically
5. Semantic Twins that enable semantic descriptions of Digital Twins and the related real-world entities
6. a Decentralised Interledger Bridge (DIB) that enables transactions across different distributed ledgers (DLTs)
7. a privacy-preserving Verifiable Credential based decentralised on-device access control solution for constrained IoT Devices
8. a QR (Quick Response) code and GS1 (Global Standards 1) Digital Link based discovery mechanisms

The solutions have been packaged for easy deployment to achieve the goals of the work package.

This document also describes a demo being developed in WP7 for the Smart City LL for the configuration of IoT devices to showcase how the integration of solutions 5-8 works. Altogether 6 out of 10 LL Use Cases need at least two of the technologies that are presented in this deliverable, in particular SSI technologies, thus motivating their importance and enabling extensive validation of the solutions.

The verification results of these solutions will be reported in the upcoming Deliverable D6.3 and the validation results from the IoT-NGIN Living Labs will be reported in Deliverable D7.4.

10 References

- [AAS] Asset Administration Shell Specifications, Plattform Industrie 4.0. 23/11/2020
<https://www.plattform-i40.de/IP/Redaktion/EN/Standardartikel/specification-administrationshell.html>
- [Ala2021] R. Ala-Laurinaho, "API-based Digital Twins - Architecture for Building Modular Digital Twins Following Microservices Architectural Style," Doctoral dissertation, Aalto University, 2021.
<http://urn.fi/URN:ISBN:978-952-64-0594-0>
- [Arm2017] Armin Haller et al. Semantic Sensor Network Ontology. Technical Specification OGC 16-079. W3C & OGC, Oct. 17, 2017. url:
<https://www.w3.org/TR/vocab-ssn/>
- [Aut2021a] J. Autiosalo, "Discovering the Digital Twin Web - From singular applications to a scalable network," Doctoral dissertation, Aalto University, 2021.
<http://urn.fi/URN:ISBN:978-952-64-0621-3>
- [Aut2021b] J. Autiosalo, J. Siegel, K. Tammi "Twinbase: Open-Source Server Software for the Digital Twin Web", *IEEE Access*, vol 9, pp. 140779-140798, 2021.
<https://doi.org/10.1109/ACCESS.2021.3119487>
- [Ber2006] T. Berners-Lee "Linked Data" <https://www.w3.org/DesignIssues/LinkedData>
- [Brei2007] K. Breitman, M.A. Casanova, and W. Truszkowski, *Semantic Web: Concepts, Technologies and Applications*, 2007.
- [But2016] V. Buterin, "Chain interoperability," R3 Research Paper, 2016.
- [Cyber2022] Cyber Resilience Act 2022, Proposal for a regulation of the European parliament and of the council on horizontal cybersecurity requirements for products with digital elements and amending Regulation (EU) 2019/1020
<https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act>
- [DTDL] Digital Twins Definition Language
<https://github.com/Azure/opensdtwins-dtdl>
- [D1.1] IoT-NGIN D1.1 - Definition analysis of use cases and GDPR Compliance
<https://iot-ngin.eu/index.php/deliverable/>
- [D1.2] IoT-NGIN D1.2 - IoT meta-architecture, components and benchmarking
<https://iot-ngin.eu/index.php/deliverable/>

- [D3.1] IoT-NGIN D3.1 - Enhancing deep learning / reinforcement learning <https://iot-ngin.eu/index.php/deliverable/>
- [D3.2] IoT-NGIN D3.2 - Enhancing Confidentiality preserving federated ML <https://iot-ngin.eu/index.php/deliverable/>
- [D3.3] IoT-NGIN D3.3 - Enhanced IoT federated deep learning/ reinforcement ML <https://iot-ngin.eu/index.php/deliverable/>
- [D5.1] IoT-NGIN D5.1 - Enhancing IoT Cybersecurity <https://iot-ngin.eu/index.php/deliverable/>
- [D5.2] IoT-NGIN D5.2 - Enhancing IoT Cybersecurity (Update) <https://iot-ngin.eu/index.php/deliverable/>
- [D5.3] IoT-NGIN D5.3 - Enhancing IoT Data Privacy & Trust <https://iot-ngin.eu/index.php/deliverable/>
- [D5.4] IoT-NGIN D5.4 - Enhancing IoT Data Privacy & Trust (Update) <https://iot-ngin.eu/index.php/deliverable/>
- [D6.2] IoT-NGIN D6.2 - Integrated IoT-NGIN platform & laboratory testing results <https://iot-ngin.eu/index.php/deliverable/>
- [D7.3] IoT-NGIN D7.3 - Living Labs use cases intermediate results <https://iot-ngin.eu/index.php/deliverable/>
- [Foa2022] Foaf Vocabulary Specification <http://xmlns.com/foaf/0.1/>
- [Fot2022] N. Fotiou, et al. "Capabilities-based access control for IoT devices using Verifiable Credentials." 2022 IEEE Security and Privacy Workshops (SPW). IEEE, 2022. <https://doi.org/10.1109/SPW54247.2022.9833873>
- [Geo2022] Basic Geo (WGS84 lat/long) Vocabulary <https://www.w3.org/2003/01/geo/>
- [Gua2009] Nicola Guarino, Daniel Eberle, and Steffen Staab. Chapter: "What is an ontology?" in Handbook on ontologies pp. 1-17, 2009.
- [Has2019] M. U. Hassan, M. H. Rehmani, and J. Chen, "Privacy preservation in blockchain based IoT systems: Integration issues, prospects, challenges, and

future research directions," *Future Generation Computer Systems*, vol. 97, pp. 512–529, 2019.

- [Jac2020] M. Jacoby, and T. Usländer, "Digital Twin and Internet of Things—Current Standards Landscape," *Applied Sciences* 2020, 10, 6519.
<https://doi.org/10.3390/app10186519>
- [LOG4J] M. Ahmed, log4j-scan, Github, 2021. <https://github.com/fullhunt/log4j-scan>
- [Mat2022] J. Mattila, R. Ala-Laurinaho, J. Autiosalo, P. Salminen, and K. Tammi, "Using Digital Twin Documents to Control a Smart Factory: Simulation Approach with ROS, Gazebo, and Twinbase," *Machines* 2022, 10(4), 225.
<https://doi.org/10.3390/machines10040225>
- [MITRE] MITRE, "CWE List Version 4.10", 2021. <https://cwe.mitre.org/data/index.html>
- [NGSI-LD] Duncan et al., "NGSI-LD API: for Context Information Management ," ETSI White Paper No. 31, 1st ed., 2019.
https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp31_NGSI_API.pdf
- [Odata] Open Data Protocol
<https://www.odata.org/>
- [One2019] oneM2M Partners Type 1. oneM2M TS-0012-V3.7.3 - Base Ontology. Technical Specification TS 103 264. oneM2M, Feb. 28, 2019. Url:
<https://www.onem2m.org/technical/onem2m-ontologies>
- [Ont2022] Vocabularies for Ontologies
<https://www.w3.org/standards/semanticweb/ontology>
- [OWL-W3] OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (Second Edition)
<https://www.w3.org/TR/owl2-syntax>
- [Psy2023] K. Psychogyios, T.-H. Velivassaki, S. Bourou, A. Voulkidis, D. Skias, and T. Zahariadis, "GAN-Driven Data Poisoning Attacks and Their Mitigation in Federated Learning Systems," *Electronics*, vol. 12, no. 8, p. 1805, Apr. 2023, doi: 10.3390/electronics12081805.
- [Saref2022] Smart Applications REference Ontology, and extensions
<https://saref.etsi.org/>
- [SAREF4CITY] SAREF extension for Smart City <https://saref.etsi.org/saref4city/>

- [SOF2020] SOFIE Project Use Cases: Context-aware Mobile Gaming Pilot <https://media.voog.com/0000/0042/0957/files/sofie-onepager-gaming-noScreens.pdf>
- [SOF2021] SOFIE project: Secure Open Federation for Internet Everywhere," <https://www.sofie-iot.eu/>, (Accessed on 03/12/2021).
- [SOSA] SOSA Ontology https://www.w3.org/2015/spatial/wiki/SOSA_Ontology
- [STA] OGC SensorThings API <https://www.ogc.org/standards/sensorthings>
- [TB-SSI-API] Semantic Twin SSI API <https://github.com/IoT-NGIN/twinbase-ssi-api>
- [TwinbaDLT] Twinbase with DLT integration <https://github.com/IoT-NGIN/twinbase-dlt>
- [Twinbase] Twinbase: Server software for hosting digital twin documents <https://github.com/twinbase/twinbase>
- [WASC] Web Application Security Consortium, "The WASC Threat Classification v2.0", 2010. <http://projects.webappsec.org/w/page/13246978/Threat%20Classification>
- [Wen2019] Wenbin lin et al. Review of Standard Ontologies for the Web of Things
- [WGS84voc] WGS84 Geo Positioning: an RDF vocabulary https://www.w3.org/2003/01/geo/wgs84_pos#
- [WoT-TD] Web of Things (WoT) Thing Description. W3C Recommendation 9 April 2020 <https://www.w3.org/TR/wot-thing-description/>
- [Wu2021] Wu, L., Kortessniemi, Y., Lagutin, D., & Pahlevan, M. (2021, September). The Flexible Interledger Bridge Design. In 2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS) (pp. 69-72). IEEE.
- [ZAP] OWASP Zed Attack Proxy (ZAP), zapproxy.org
- [Zha2019] J. Zhang, S. Zhong, J. Wang, L. Wang, Y. Yang, B. Wei, and G. Zhou, "A review on blockchain-based systems and applications," in International Conference on Internet of Vehicles. Springer, 2019, pp. 237–249.