D2.2

The COLLABS Level-3 Security Package for Secure Digital Supply Networks: 1st complete version

Project number:	871518
Project acronym:	COLLABS
Project title:	A COmprehensive cyber-intelligence framework for resilient coLLABorative manufacturing Systems
Start date of the project:	1st January 2020
Duration:	36 months
Programme:	ICT-08-2019

Deliverable type:	Other
Deliverable reference number:	DS-01-871518/ D2.2 / v1.0
Work package contributing to the deliverable:	WP 2
Due date:	JUN 2021 - M18
Actual submission date:	JUN 2021

Responsible organisation:	HUA
Editors:	Dr. Panagiotis Rizomiliotis Dr. Konstantinos Tserpes Mrs. Aikaterini Triakosia
Dissemination level:	PUBLIC
Revision:	FINAL

Abstract:	*This the second deliverable of Work Package 2 tasks T2.1 (Tools and methods for secure data sharing), T2.2 (Trustworthiness of data flows), T2.3 (Machine learning-based cognitive security framework), T2.4 (Statistical Analytics and Machine- / Deep-Learning on shared data), T2.5 (Distributed anomaly detection for Industrial IoT) and T2.6 (Workflow-driven security for supply chain and compliance in manufacturing) related to the 1st version of integrated platform of the project. It describes and demonstrates the various technologies that form the COLLABS Level-3 security package for secure digital supply networks.*
Keywords:	*Secure data sharing, secure data flow, anomaly detection, statistical analytics, IIoT, digital supply chain*

The project COLLABS has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871518.

Editors

Dr. Panagiotis Rizomiliotis (HUA)
Dr. Konstantinos Tserpes (HUA)
Mrs. Aikaterini Triakosia (HUA)

Contributors *(ordered according to beneficiary numbers)*

Idryma Technologias Kai Erevnas, FORTH
Advanced Laboratory on Embedded Systems SRL, ALES
University of Novi Sad Faculty of Sciences, UNSPMF
Sphynx Technology Solutions AG, STS
Thales Six GTS France SAS, TSG
Information Technology for Market Leadership, ITML
Universita Degli Studi Di Padova, UNIPD
Siemens AG, SAG
Renault SAS, REN
Harokopio University, HUA

Document Revisions & Quality Assurance

Internal Reviewers

1. *Ernesto Gomez Marin, IFAG*
2. *Alzahraa Alhaddad, ITML*

Revisions

Version	Date	By	Overview
1.0	15/6/2021	HUA	Final version
0.5	11/6/2021	Internal Reviewers	Internal Review
0.4	31/5/2021	HUA	Editing
0.3	25/5/2021	Partners	Second round of inputs
0.2	15/5/2021	Partners	First round of inputs
0.1	12/03/2021	Editors	ToC

Disclaimer

Executive Summary

The COLLABS project aims at developing, demonstrating, and supporting a comprehensive cyber-intelligence framework for collaborative manufacturing. This enables the secure data exchange across the digital supply chain while providing high degrees of resilience, reliability, accountability, and trustworthiness. It addresses threat prevention, detection, mitigation, and real-time response.

This deliverable provides a specification of the components and supporting tools and technologies in the ***1st version of integrated platform*** *of the COLLABS Level-3 security package for secure digital supply networks. These components are implemented through activities within the six tasks of work package 2. This third level of security mechanisms (1) comprises tools for secure data sharing, ensuring trustworthiness of data flows in collaborative manufacturing environments, (2) delivers a machine learning-based cognitive security framework applied to shared data, and (3) deploys a workflow-driven security framework for supply chain and compliance in manufacturing based on distributed ledgers.*

After the introduction, in Section 2 we give an overview of the aspects of the COLLABS architecture addressed by Level-3 security, and we report the differences between the MPV and the 1st version of integrated platform. Then, in Section 3, we describe the components of the COLLABS framework that implement Level-3 security for the 1st version of integrated platform. In Section 4, we give an overview of the architectural framework for trustworthiness assurance. Section 5 concludes the deliverable. Finally, in the Appendix, we present supporting methods that have been used in the design and development of COLLABS components.

Table of Contents

List of Figures

List of Tables

1. Introduction

Security within the COLLABS framework is organized into three levels, namely: Level 1 – hardware-enabled and device-level security, Level 2 – Inter-device level security based on distributed ledger technologies, and Level 3 – Machine learning-based cognitive security level. The aim of this deliverable is to provide an overview and description of components and supporting tools and technologies relevant to the 1st version of integrated platform implementation of the COLLABS Level-3 security package for secure digital supply networks.

The identified objectives that the COLLABS Level-3 security package needs to achieve are providing tools for secure data sharing, ensuring trustworthiness of data flows in collaborative manufacturing environments, delivering a machine learning-based cognitive security framework applied to shared data, and deploying a workflow-driven security framework for supply chain and compliance in manufacturing based on distributed ledgers. For more details about the objectives of COLLABS Level-3 security please see deliverable D1.3.

Work on the COLLABS Level-3 security package permeates all tasks of Work Package 2:

T2.1, Tools and methods for secure data sharing, deals with the required hardware and software architectures. It introduces the notion of trusted execution environment (TEE) and explains the concepts relevant to TEE functional availability in a device. The task addresses methods to remotely and dynamically manage TEEs, safe means to facilitate multiparty analytics of sensitive data, as well as security of the data from unauthorised access, and the anonymisation of the data. More details about the work done within this task are given in Section 3.2.

T2.2, Trustworthiness of data flows, addresses the detailed design and development of the building blocks composing the trust infrastructure. To achieve this, the components and results developed in multiple other tasks are leveraged and combined, including machine learning, deep learning, and anomaly detection, with more details given in Section 4 of this deliverable.

T2.3, Machine learning-based cognitive security framework is concerned with developing the core of the Level-3 security mechanism. This includes behavioural models that will enable the analysis of the network flows among multiple IoT devices which is supported by methods for device identification and wireless fingerprinting based on features derived from packet headers as opposed to packet payloads, thus, facilitating use on both encrypted and non-encrypted network traffic. More details about the work done on this framework are given in Section 3.1.

T2.4, Statistical analytics and machine- / deep-learning on shared data, will offer privacy-preserving storage and analytics to end-users and enterprises by allowing seamless integration and injection of heterogeneous data, and facilitate the adoption of collaborative analytics to enterprises, without exposing private or sensitive information. More details about the work done within this task is given in Section 3.2.

T2.5, Distributed anomaly detection for industrial IoT, advocates the introduction of intrusion detection systems (IDS) which are not limited to the local view of the device or a group of devices they are installed on, but rather employ a lightweight and distributed approach that leverages innovative machine-learning techniques to detect specific security threats and to identify malicious behaviours. The work on this task is addressed in Section 3.4 of this deliverable.

T2.6, Workflow-driven security for supply chain and compliance in manufacturing, provides a workflow framework for security and resilience in collaborative manufacturing, allowing users to organise production processes across companies in a secure manner. To achieve this objective, the task implements the formalisation of contractual obligations and of security and compliance requirements. It integrates data streams from trustworthy sensors and provides concepts for enforcing the workflows in the secured collaborative manufacturing supply chain, leveraging technologies such as distributed ledgers, blockchain, and distributed databases. Section 3.3 of this deliverable is relevant to this task.

Within the COLLABS trust infrastructure, the components and tools that facilitate data protection, developed in WP2, will consider both, data in transit and data at rest. An important part of the infrastructure is the definition of an encrypted data flow model that specifies the use of cryptographic primitives, enabling data from IoT devices to be securely transferred to and processed by trusted execution devices or end cloud-services.

The rest of the deliverable is structured as follows.

- Section 2 gives an overview of the aspects of the overall COLLABS architecture that are addressed by Level-3 security for the 1st version of the integrated platform. Also, it presents the differences between this version and the MVP.
- Section 3 describes the components of the COLLABS framework that facilitate Level-3 security, relevant to the 1st version of the integrated platform, which encompasses the machine learning-based cognitive security framework, statistical analytics and machine- / deep-learning on shared data, computations on outsourced data, workflow-driven security for supply chain and compliance in manufacturing and network traffic monitoring.
- Section 4 describes the architectural framework for trustworthiness assurance and its main building blocks: Secure elements, data protection, and encrypted traffic analysis. Within the topic of data protection, details are provided on the data flow model specification, as well as specific associated scenarios provided by three project partners (initially described in deliverable D1.3), namely by ALES (S1 - controlled and secure remote maintenance, S2 - controlled share of compliance data, S3 - trusted compliance data share across the supply chain, S4 - analysis of manufacturing performance on a global scale), PCL (S1 - shop floor threat detection and prevention, S2 - remote data sharing), and REN (S1 - controlled and secured remote maintenance, S2 - cloud-based architecture for industrial processes, S4 - security of connected devices).
- Section 5 concludes the deliverable summarizing the work that has been done and presenting the next steps.

The Appendix presents the supporting methods used for screening and improving the security of the COLLABS solutions. It includes methods and guidelines for oblivious machine learning, distributed anomaly detection for industrial IoT, and the formal specs verifier (FSV). This section aims to make this document as self-contained as possible.

2. Level-3 Security Components Relevant to the 1st Version of the Integrated Platform of the COLLABS Architecture for Secure Digital Supply Networks

The third level of security of the COLLABS Architecture for Secure Digital Supply Networks (explained in more detail in Section 1 and deliverable D1.3) includes the components that have the ability to supervise information exchange between the devices either within the boundaries of the smart factory, or with external partners within the digital supply network. By monitoring various data flows from the connected devices and creation of situational awareness they will be capable of anticipating security risk situations and activating prevention mechanisms. According to technology and mechanisms, the Level-3 security components could be divided into four main functional groups:

1. *Group 1: Machine Learning-Based Cognitive Security Framework* - aims to develop behavioural models that will enable the analysis of the network flows among multiple IoT devices. It amalgamates the components: IoT Secure Wireless Fingerprinting and Machine Learning Structured (Non) Convex (ML S(N)C) Optimization, and GMS Tools. The first one is based on scanning fingerprint information of multiple IoT devices, while the second one uses Machine Learning (ML) / Deep Learning (DL) models that enable the analysis of their behaviour.
2. *Group 2: Statistical Analytics and Machine- / Deep-Learning on Shared Data* - will offer integration of heterogeneous data of the end-users and enterprises and facilitate the adoption of collaborative analytics to enterprises without exposing private or sensitive information. Homomorphic Encryption will be the main cryptographic technology used for the secure computation outsourcing. It, also, presents the synergy of the components named Security Infusion and 3ACEs.
3. *Group 3: Workflow-Driven Security for Supply Chain and Compliance in Manufacturing* - will provide a workflow framework based on distributed ledger technology for security and resilience in collaborative manufacturing.
4. *Group 4: Network Traffic Monitoring* – will provide tools for monitoring the behaviour of a network. The first tool will monitor encrypted traffic, while the second will detect anomaly behaviour using a decentralized approach.

What is new in the 1st version of the platform

All the components have been developed, integrated into the COLLABS Architectural Framework and made ready for presentation. In order facilitate the monitoring of each component's progress in the project, the new subsection "Component Evolution" has been added in Section 3 per component.

Some of the components that appear in this 1st version of COLLABS platform were initially integrated into the first minimum viable product of the project. More precisely, two components in Group 2, Security Infusion and 3ACEs, and the component in Group 3, the Workflow-Driven Security Framework, have been already presented as part of the MVP. Of course, the new versions of all these components offer further functionalities, as it is analysed in the corresponding "Component evolution" subsection in each component.

The two components in Group 1 were initially created during the first year of the project, but only initial functionalities were initialized. The enhanced version of the two components is now integrated into the 1st version of the platform.

Finally, we have three new components. Namely, the two components in the 4th group, the Network Traffic Monitoring group, and the Homomorphic Encryption component in the 2nd group. All three components were designed during the first year. The methodology used for these designs was analysed in deliverable D2.1 and it appears in the appendix of the current document.

3. Security Component Descriptions

3.1 Machine Learning-Based Cognitive Security Framework

The Machine learning-based cognitive security framework consists of two subcomponents, IoT secure wireless fingerprinting (Section 3.1.1) and ML S(N)C optimization and GMS tools (Section 3.1.2). The two components work together by the former component providing data (device fingerprints), and the latter producing machine-learning models capable of identifying devices and detecting anomalies. The envisioned interaction between the two subcomponents within the COLLABS framework is illustrated in Figure 1.

3.1.1 IoT Secure Wireless Fingerprinting

This component will provide inputs to the Machine learning-based cognitive security framework in the form of auxiliary information referred to as fingerprints, which are collected from wireless Industrial IoT devices and delivered along with the sensor data to the network server. Fingerprints represent all information that can be acquired from an IIoT device which provides information about that device and its usual behaviour. Examples of fingerprint information include radio channel condition parameters, hardware-related imperfections attributable to a particular IIoT device, detailed energy consumption information, or similar. Based on the fingerprinting inputs, ML algorithms will be trained to differentiate between normal and abnormal identities of IIoT devices and initiate appropriate actions.

Functional Description

Depending on the wireless interface used by the IIoT device, this component will collect information about the radio-channel fingerprints from the device communication module and process this information into a useful stream of fingerprinting data. Use cases considered in the project will start with most common wireless IoT technologies such as IEEE 802.11ah or similar (Wi-Fi IoT) and 3GPP NB-IoT (Cellular IoT). For the emerging IEEE 802.11ah Wi-Fi IoT standard, we will use experimental platforms based on software-defined radios to generate and collect fingerprinting data. For IEEE 802.11ac standard, which is one of the commonly used broadband Wi-Fi interface used today, we will collect channel state information from commercial 802.11ac devices or access points. This imposes certain requirements, for example, in terms of the 802.11 chipset the device or access point is based on, as it is not possible to extract Channel State Information (CSI) as fingerprints for any available 802.11ac device[1]. For 802.11ac, we will also collect CSI fingerprints from Raspberry Pi devices, which are now commonly used as sensor platforms in industrial IoT. Finally, we will also collect fingerprinting data in the form of radio channel parameters (RSSI, RSRP, SNR) from 4G Cellular IoT devices based on 3GPP NB-IoT technology which is the main standard for 4G/5G cellular IoT connectivity. Initial phase of fingerprinting collection will be done in lab environment (UNSPMF). Such data sets will be used as a proof of concept for training relevant machine learning models. Further experiments will be done based on the data available at the use case

[1] *CSI is estimated at the lowermost physical layer of IEEE 802.11ac and this information is used to perform received signal equalization and correctly decode received data packets. However, this information is not forwarded to higher layers and standard commercial Wi-Fi device has no access to this information from Wi-Fi chipset. However, certain Wi-Fi chipsets (e.g., some Intel and some Atheros/Qualcomm chipsets) allow extraction of this information using appropriate drivers.*

providers. For example, radio channel conditions collected from LoRa devices and Wi-Fi nodes (provided by REN) will be considered as suitable fingerprints.

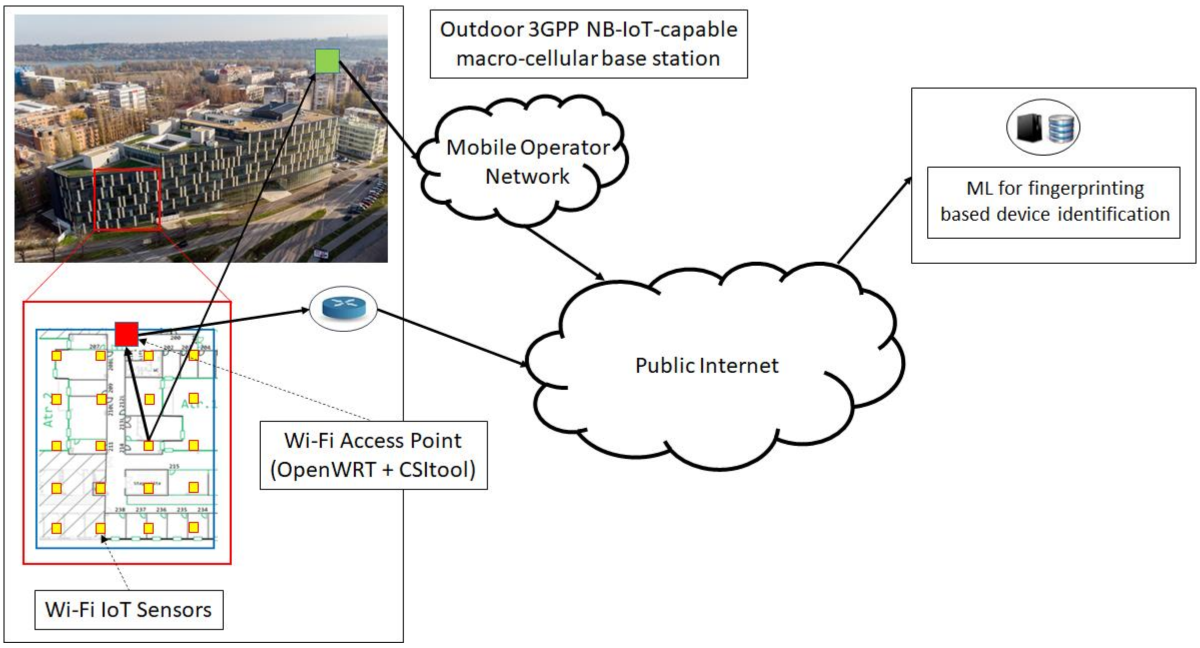

Outdoor 3GPP NB-IoT-capable macro-cellular base station
Mobile Operator Network
ML for fingerprinting based device identification
Public Internet
Wi-Fi Access Point (OpenWRT + CSItool)
Wi-Fi IoT Sensors

Figure 1 IoT wireless fingerprinting setup.

Preconditions and Input

Most of the commercially available IIoT devices are currently not suitable for the extraction of appropriate fingerprints from the device itself. Although most of the devices currently deployed at the use-case provider premises will not be useful for demonstration purposes, some possibilities for extracting channel state information from existing devices have been identified. For example, specific LoRa devices for outdoor smart logistics applications, as well as indoor Wi-Fi based IoT devices, provide possibilities for logging and using specific channel state information and device information data as fingerprints. Another possibility of using UNSPMF devices developed under lab conditions at UNSPMF premises and transferred to a real-world deployment at use case providers, will remain open.

Postconditions and Output

There are no specific postconditions. Once the IIoT device is capable of collecting fingerprints, it will send fingerprints formatted as other usually transmitted data from other onboard sensors, along this same data, in order to efficiently use communication opportunities. Thus, fingerprinting data can be considered as data coming from additional sensors.

Output of the component will be formatted in a way which is appropriate for interpretation by ML- based cognitive security framework. For example, fingerprinting data may be delivered to the ML-based cognitive security framework as a feature vector representing average CSI values measured from the signals arriving at each of the Wi-Fi device/access point antennas and averaged across a sequence of a given number of last received Wi-Fi packets.

Component Design

Depending on the wireless interface, fingerprinting data can be collected in different ways. For Wi-Fi-based technologies, specific software tools are developed that need to be integrated within Wi-Fi access points and/or Wi-Fi IoT devices in order to gain access to fingerprinting information[2]. This will be first tested for the most common IEEE 802.11ac devices and access points for which there exists a solution for the extraction of CSI-based fingerprinting information (although not for every 802.11ac chipset). Besides open source tools for extracting fingerprints for IEEE 802.11ac devices, it is also possible to use proprietary tools used for management and control of Wi-Fi devices already available at the use case provider for collection of fingerprints.

For 3GPP Cellular IoT devices, the device is already exchanging a lot of radio-interface parameters with the network, many of which could be captured at the IoT device and transmitted as auxiliary data back to the network servers and the ML S(N)C optimization and GMS tools component. We are also capable of designing and fabricating NB-IoT devices which could be customized for specific types of fingerprinting data collection. Our current NB-IoT testbed already possesses about 100 custom-designed NB-IoT devices which will be readily available for large-scale data collection purposes. In addition, it is possible to use proprietary tools available at use case provider for LoRa low-power wide area network devices, that offers logging various channel quality and device status parameters that can be used as fingerprints for further machine learning based processing.

Addressed COLLABS Common Security Requirements

List of security requirements addressed by this component:

- CSR_01 - Identification and Authentication: This requirement is partially addressed by wireless fingerprinting component as it represents a crucial input for fingerprinting-based authentication and device identification.
- CSR_08 - Monitoring: This requirement is partially addressed by wireless fingerprinting as the input from wireless fingerprinting will serve as a data for IIoT device monitoring and anomaly detection.

Beyond the state of art

Wireless fingerprinting is a recent and popular research area in wireless signal processing. In COLLABS, we expanded the investigation of extracting wireless fingerprints to two novel communication standards that has not been addressed in the literature, namely, IEEE 802.11ah Halow (emerging low-power Wi-Fi for IoT) and 3GPP NB-IoT. For the latter, in-house designed nodes are used that are specifically designed for efficient fingerprinting acquisition. Initial usage of extracted wireless fingerprints in COLLABS is for the problem of device identification. Device identification using wireless fingerprints has been addressed recently, but only in limited studies and with limited signals. In COLLABS, we extend device identification to IEEE 802.11 and 3GPP NB-IoT technologies and use a variety of machine learning methods to address this problem.

Component Evolution

Wireless fingerprinting component extracts additional information about wireless IoT devices that is used by machine learning methods to learn a new knowledge about the device. The work on the component was started within the COLLABS project. Up to D2.1 and the MVP, two data sets based on IEEE 802.11ah Wi-Fi IoT and IEEE 802.11ac technology were generated. Since then, extraction of wireless fingerprints from 3GPP NB-IoT devices was conducted and third data set was generated. Now, we are in the process

[2] *Currently, there are several tools available for integration with Wi-Fi devices. For example, Atheros CSI Tool (https://wands.sg/research/wifi/AtherosCSI/) can be used to extract this information from Atheros/Qualcomm-based chipsets.*

of extracting wireless fingerprints from IEEE 802.11ac based Raspberry Pi devices which are wide-spread platform for industrial IoT. Currently, efforts are underway to integrate the wireless fingerprinting component into LoRa devices in wide area LoRa network and IEEE 802.11ac devices in industrial setup in collaboration with REN.

Outlook

The initial version of the component will involve developing appropriate Wi-Fi and Cellular IIoT devices and collecting the corresponding data sets for training the ML-based cognitive security framework. The deployment and testing will be done exclusively at UNSPMF premises. In the further stages, the solutions for integration of both the IIoT devices and the ML S(N)C optimization and GMS tools component will be investigated in collaboration with the use case providers.

3.1.2 ML S(N)C Optimization and GMS Tools

The component is organized around a framework consisting of Machine Learning (ML) and Deep Learning (DL) models that enable the analysis of the behaviour of multiple IoT devices, as well as application to other COLLABS-related problems. For instance, data collected from the IoT secure wireless fingerprinting component will be fed into this component. The way input data (device signals) interact with parts of the component to produce the desired output and the evaluation reports is illustrated in Figure 2.

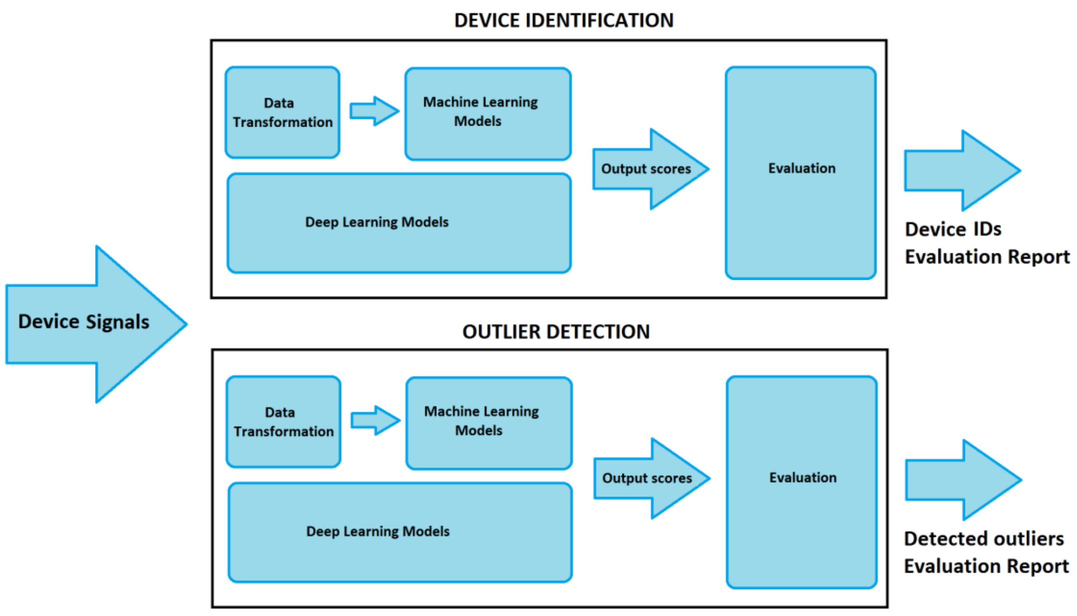


Figure 2 IoT wireless fingerprinting setup.

Models can be built in a supervised or unsupervised way depending on the application domain and available data. If class labels are present (e.g., device IDs), supervised ML and DL methods can be trained to solve a classification problem (e.g., device identification based on various fingerprint features) or outlier detection problem (e.g., detecting faulty or compromised devices). If the data consists only of normal instances, then ML models based on a semi-supervised anomaly detection approach also known as novelty detection can be applied. On the other hand, if class labels are not present, unsupervised methods for outlier detection can be applied.

Functional Description

Modules of this component implement models for classification and anomaly detection (AD) that can be applied to fingerprint data, which include supervised methods based on support vector machines (SVMs), random forests, naive Bayes, and different architectures of deep neural networks, as well as semi-supervised (novelty detection) and unsupervised outlier detection methods. Models can be trained in a supervised, semi-supervised or unsupervised way (depending on the availability of labels in collected data) to identify individual devices and/or detect anomalous device behaviour based on their fingerprints.

The functionality of the component is not limited to device fingerprinting, identification, and behaviour analysis; it can be adapted to any problem domain.

Preconditions and Input

- Preconditions: environment (native OS or VM) able to execute Python and associated libraries (scikit-learn, PyTorch, PyOD, LightGBM, Sktime, Pandas, SciPy, Seaborn, NumPy, Matplotlib, Missingno).
- Sufficient processing power and RAM for model training and deployment, depending on the problem at hand and employed data. GPU acceleration is recommended for training DL models. All models can be executed on a CPU, while DL models are far more efficient if executed on a GPU. Parallelization is supported by some methods, which means that more CPU cores can contribute to efficient model prediction. There are various types of models that can fit in different memory resources in compromise with the accuracy of the model. Current trained DL models can run on a GPU that has at least 4GB of memory.
- The component currently supports input of data in the form of .csv files; other means of input (e.g., direct network read) will be added as dictated by the project's needs.

Postconditions and Output

- There are no specific postconditions.
- The component currently supports output in the form of plain text and .csv files; other means of output will be added as dictated by the project's needs.

Component Design

The component consists of modules for:

- Data input and processing.
- Supervised machine learning (classification) by classical (non-DL) methods, including:
    - Support vector machines (SVM),
    - Naive Bayes (NB),
    - Decision trees (DT),
    - Random forest (RF),
    - Gradient boosting machines (GBM).
- Supervised deep learning, including:
    - Fully connected neural networks (FNN),
    - Convolutional neural networks (CNN),
    - Recurrent neural networks (RNN), with different variants, including:
        - Long short-term memory networks (LSTM),
        - Gated recurrent unit (GRU).
- Unsupervised and semi-supervised outlier/anomaly detection including methods based on:

  - One-class support vector machines (OCSVM),
  - Isolation forests,
  - Local outlier factor (LOF),
  - Angle-based outlier detection (ABOD),
  - k-nearest neighbours (kNN),
  - Elliptic envelope,
  - Histogram-based outlier score,
  - Copula-based outlier detection,
  - Autoencoders.
- Model evaluation:
  - Training/validation/test accuracy,
  - Training/test speed.

The component is written in Python and depends on external libraries scikit-learn, PyTorch, PyOD, LightGBM, Sktime, Pandas, SciPy, Seaborn, NumPy, Matplotlib, Missingno. Parallelization is supported by many methods.

Addressed COLLABS Common Security Requirements

- CSR_01 - Identification and Authentication: partially because the component introduces one of the means for device identification that can support classical methods for authentication and identification.
- CSR_08 – Monitoring: partially because the component provides one of the means for determining possible security breaches by detecting anomalous behaviour.

Beyond the state of art

The advances with respect to the state-of-the-art are expected through innovative applications of ML and OD techniques in the domains of device identification and anomaly detection using novel to obtain and extract wireless fingerprinting data from IoT devices. Coupled with the novel approaches to wireless fingerprinting introduced by the IoT Secure Wireless Fingerprinting component, we expect the ML S(N)C Optimization and GMS Tools component to provide extended security functionality compared to standard approaches to device authentication and threat detection.

Component Evolution

The development of the component was started within the COLLABS project. Up to D2.1 and the MVP, multiple techniques for classification and outlier detection were implemented and evaluated on two data sets generated at the UNSPMF lab premises by the IoT Secure Wireless Fingerprinting component, and the results presented as an MVP demonstrator. Since then, the component was extended with:

- New machine learning methods related to time-series (such as different variants of LSTMs),
- New deep learning methods for outlier detection (different types of autoencoders),
- New semi-supervised outlier detection methods (such as copula-based outlier detection)

In addition, experiments with existing methods were repeated on the third data set produced by the UNSPMF Wireless Fingerprinting component (NB-IoT), and new methods were included in the experiments on all three UNSPMF data sets. Currently, efforts are underway to integrate the component into at least two COLLABS use-case scenarios.

Outlook

The initial version of the component was applied to the problems of device identification and threat detection in conjunction with the IoT Secure Wireless fingerprinting component. The deployment and testing in this phase were done exclusively at UNSPMF premises. Currently and in the further stages, the solutions for integration of both the IIoT devices and the ML S(N)C optimization and GMS tools component are being investigated in collaboration with the use case providers. In addition, the component will be used to address the task of Confidential Data Discovery.

3.2 Statistical Analytics and Machine- / Deep-Learning on Shared Data

3.2.1 Security Infusion

This component is agent-based software which is used to monitor and secure any network's infrastructure. Security Infusion offers real-time and historical data access related to the collected, analysed, and visualised data concerning the security posture of a company. Security Infusion was designed based on ITML's experience of managing IT resources, and operational risk. The service's main aim is to do the job in a simple and efficient manner, without compromising the performance of the protected resource or accuracy of the data and information it collects regarding event and processes.

Functional Description

Security Infusion is a cloud-based, online service, mainly used for information security and event management of a network, which contains IDS functionalities as well. In the framework of the COLLABS pilots, the cloud might not be a valid option for some use cases, therefore edge deployment will be used as a substitute. In industrial environments, the component can be used to detect and mitigate threats on a multitude of vulnerable IoT Devices, by detecting unprecedented anomalies in an OT environment. Consequently, the platform creates alerts when a threat/anomaly is detected. It employs forensic analysis features which store information related to past logs and events, which administrators can access when needed. Security Infusion also delivers thorough vulnerability assessments and port scans to assess certain future issues and prepare administrators to avoid and/or resolve them.

Preconditions and Input

- The Infusion edge manager differs in preconditions depending on the number of agents:
  - For up to 20 agents, an AMD64 processor is needed, along with 2 CPUs and a 4G memory, where 75KB are needed per snapshot of the system.
  - For up to 100 agents, an AMD64 processor is needed, along with 2 CPUs and a 6G memory, where 75KB are needed per snapshot of the system.
- Consequently, the requirements differ depending on the OS type:
  - For the windows agent, a minimum of windows 7 OS, a 32/64bit, an intel i3-i5, and a 3G-4G memory are required.
  - For the Linux agent, a CentOS, or Ubuntu 18.04 OS, a 32/64bit, a JRE 1.8, an intel i3-i5, and a 3G-4G memory are required.
- Security infusion processes the following data:
  - Inputs system data, OS data and network data and syslog events,
  - Internet access to send data to the cloud is needed.

Postconditions and Output

- The output is mainly visual and can be saved in a Json format.
- Data processing occurs on minified / filtered data and events that the agents provide when installed.

Component Design

The service consists of three sub-tools; the Infusion Agents, the Infusion Manager (Field gateway or Cloud based), and the Infusion Web Interface. The technologies comprising the submodules of Security Infusion are delivered via Kubernetes and Docker and are comprised by a plethora of toolsets/ frameworks like Laravel[3] as the existing admin panel and dashboard, Java SDK[4] for linux-based systems' agents, Nginx[5] as a front end web layer delivering the cloud manager, and modules from the Elastic stack[6] in order to persist data and be able to perform analysis on them.

The Infusion agents are basically responsible for data collection; therefore, they are deployed in the network. The agents can be deployed o Windows and Linux operating systems, which gives users the flexibility to use them wherever necessary. Consequently, there are two types of agents, the master agent is the one responsible for data streams, port scans, vulnerability assessments, and containing log servers, whereas the data agents explicitly collect data.

The Infusion manager is responsible for a multitude of operations including data reduction, threat management, and anomaly detection reasoner. Finally, security infusion contains a friendly UI, which contains the following features:

- Dashboard,
- Event Analyzer,
- Monitoring,
- Vulnerability and Port Scan,
- Reporting,
- Admin panel.

The aforementioned features, as well as the high-level functional diagram, are shown in the following image.

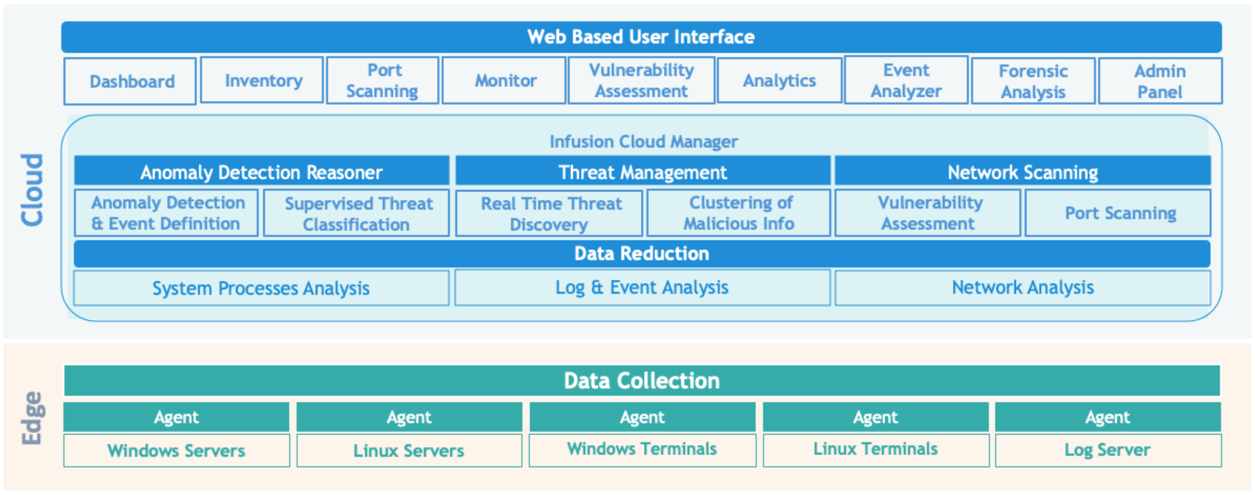


Figure 3 The high-level functional diagram

[3] Laravel PHP MVC framework, https://laravel.com/, accessed on December 2020
[4] Java SE/EE SDK, Oracle, https://www.oracle.com/java/technologies/javase-downloads.html accessed on December 2020
[5] NGinx web server, https://www.nginx.com/, accessed on December 2020
[6] Elastic stack, https://www.elastic.co/elastic-stack , accessed on December 2020

Addressed COLLABS Common Security Requirements

- CSR-08 (Monitoring) fully because the agents collect and minimize this data and send it to the cloud manager.
- CSR-09 (Accountability and non-repudiation) Fully because the forensic analysis is covered by the module by storing data for a pre-defined period in order to allow post-event forensic analysis.
- CSR-15 (Availability) as long as it includes the cloud installation of the Security Infusion cloud manager.

Beyond the state of art

Security Infusion aims to advance the state-of-the-art by providing a package of security services that efficiently safeguard devices, spanning from edge to cloud infrastructure. This aim will be achieved mainly through the following functionalities: a) provision of configurable alerts of various severity levels, through different channels to the end user, and b) mitigation of preliminary automated actions that reduce response time to intrusions. Compared to standard approaches, Security Infusion, integrated into an IIoT infrastructure, aims to significantly reduce threat detection time as well as increase responsiveness efficacy upon intrusion events.

Component Evolution

Within COLLABS, Security Infusion has already been successfully integrated and installed on small scale, testing environments. For the next stages of COLLABS, the component is planned to evolve in two steps; first, by implementing the creation and management of configurable alerts and then, by implementing threat mitigation to edge devices. The latter case may include any preliminary action that can be automatically (or semi-automatically) employed in a case of a detected intrusion, for example disabling temporarily a device, until a more sophisticated action is executed by a human operator.

Outlook

For the 1st integrated version, Security Infusion builds upon the version executed and evaluated in the context of the MVP, expanding its functionality to further engage human operators for detecting imminent threats. More specifically, all data collected by Security Infusion agents at this level are presented to the user with more sophisticated, dedicated UI, through which users are able to promptly detect abnormal behaviours and apply aggregate functions for presenting on collected data. Additionally, customised alerts notify users for high severity threats. This functionality will be further extended to include preliminary threat mitigation action.

3.2.2 3ACEs – Analytics as a Service

With the widespread use of big data analytics and its applications and the variety of data regarding size, type, and sources, industries are in a growing need of customisable tools which coincide with their changing requirements. Module Analytics as a Service (3ACES) is a tool which offers ideal decision support attributes for business through running big data analytics on whole datasets, regardless of the status which the datasets are delivered to the tool (structured, unstructured, or noisy). The data analytics features which 3ACES utilize depend heavily on machine learning algorithms which can serve in the clustering, classification, regression, and anomaly detection of the data presented.

Functional Description

The interface's flexibility along with the tool's applicability in a multitude of domains, makes it ideal to work in any business / industrial environment with the help of its machine learning algorithms. The algorithms built in the system analyses the data in 4 main steps, as shown in the figure below:

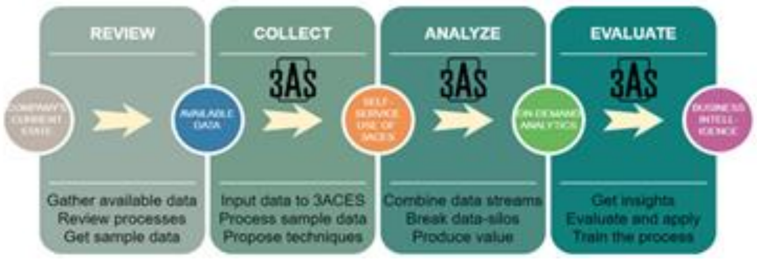


Figure 4 3ACES data analytics process.

The Data Fusion Bus (DFB) sub-module of 3ACEs can integrate several data sources related to a specified problem / use case. This data can then be analysed in a preliminary analysis using ML algorithms and assess the produced results (as defined/expected for each use case). Then several iterations with more data can enhance the accuracy of the results by further training the selected algorithm(s).

Preconditions and Input

- Onboard, the data steams as plain CSV files and/or as an Apache Kafka integration.
- MQTT events to be added within the scope of the project.

Postconditions and Output

- Provide output and respective feedback towards the end user or within the scope of the COLLABS framework to Security Infusion

Component Design

The analytics software applications created under the 3ACES framework are developed mostly as docker containers orchestrated by Kubernetes. This offers cloud optimization capabilities, allowing products based on the 3ACES framework to be deployed over public, private or hybrid cloud arrangements and enabling Software as a Service (SaaS), and/or Platform as a Service (PaaS) delivery models.

The tool also incorporates a set of ready-made open technologies, along with ITML's data fusion bus to produce a custom-made pervasive tool which both facilitates and supports businesses' decision-making processes. The open technology software includes:

- Apache Kafka data stream-processing software platform,
- KSQL streaming engine,
- Elasticsearch search engine and its stack (i.e., Logstash data streaming tool, Kibana visualization dashboard),
- Apache Spark – for programming data processing patterns, mostly using Python,
- Apache zookeeper coordination service,
- Grafana metric analytics & visualization suite,
- Laravel web programming environment.

The data fusion bus (DFB) enables the system to synchronize data from heterogeneous sources, while also providing data distribution and common data sharing interfaces to third-party services.

Addressed COLLABS Common Security Requirements

- CSR_08 (Monitoring) fully because providing the service containerized allows for OOTB monitoring tools targeted to cloud-based installations.
- CSR_14 (Availability) fully due to the backup of the cloud-native nature of the tool that allows for regular backups that ensure the protection against data loss.
- CSR_04 (Authorization), CSR_10(Integrity), and CSR_11(Integrity) all fully because all logs and data are maintained in an environment such that access is granted to specific users on a least privilege principle.

Beyond the state of art

3ACEs stands out in comparison to state-of-the-art approaches by combining efficient integration, analysis, presentation and monitoring of different data sources, in a trustworthy and scalable way. 3ACEs along with its sub-components offer various options of security including control over the KAFKA infrastructure, which is important in a big data environment by offering higher fault tolerance in comparison to traditional message brokers, supporting significantly higher throughput. Furthermore, both authentication and authorization are added through certificates which ensure that data transfer between the producers and consumers of data and the Apache Kafka is encrypted, while preventing unauthorized modules from sending on or receiving data within an IIoT framework. Finally, the component provides an easy way to manage authorized and authenticated producers and consumers of data through its API which can be used to allow or block access of modules to the data stream.

Component Evolution

Within COLLABS, 3ACEs aims to advance from TRL5 to TRL6 at the 1st integrated version of the platform, and finally to TRL7 at the final release. The added value to this component will be the result of the implementation of advanced monitoring functionalities on collected security events and related data, and its utilization in demanding real-world use cases.

Outlook

3ACES' main role will be realized following the deployment of the first MVP in most use cases to accommodate data integration between modules (per use case or for the whole COLLABS framework), giving the chance to the tool to be further developed employing the data it collects through the first pilot. Succinctly, the overall customization of 3ACES which were used in the MVP will also be realized as the requirements become more apparent post-deployment and testing.

3.2.3 Homomorphic Encryption

The HE component is used for securely outsourcing computations to the cloud. In the context of COLLABS, we aim to protect a product compliance analysis algorithm in the Scenario 2 of ALES. HE, as a cryptographic tool, aims to protect the confidentiality of the input data and/or of the computations. An algorithm is transformed into a HE algorithm by replacing the operations with HE operations. Unfortunately, this is still not an easy process and the proposed compilers are performing poorly compared to the transformation hand-crafted by a human expert. The COLLABS component was created by COLLABS experts following the guidelines that appear in the Appendix 6.1.

Functional Description

The HE component is used to:

- Encrypt client's data and protect their confidentiality. The data can stay always encrypted during their life cycle.
- Outsource these data to the cloud and perform computations on the encrypted data

More precisely, it implements an algorithm that computes the distance between a point cloud and a model, represented by a triangular mesh. It is also possible to define distance between two meshes, that it is a generalization of this use case. Practically we compute Euclidean type distances. The client, i.e. the data owner, encrypts all the points (cloud and triangle mesh) and sends them to the cloud. The cloud service provider performs the computations on the encrypted data and returns the encrypted outcome. During the whole processing, the outsourced data are always encrypted.

In more details, let C be a "point cloud" formed of NC points in the 3-dimensional space: C = {Pi | 0< i ≤ NC} and let M be a triangle mesh formed of NM triangles M = {Ti | 0<i≤ NM }. C can be represented as a NC x 3 matrix which values are real numbers and each triangle is described 3 points in the 3-dimensional space.

The HE process appears in the following *Figure 5*.

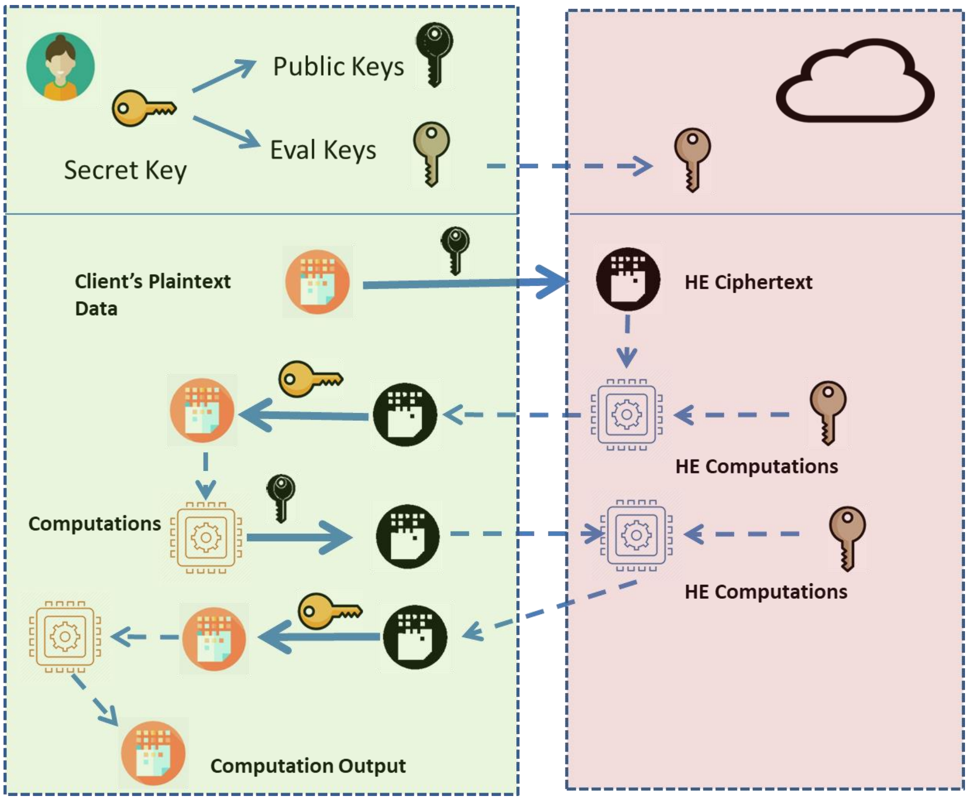


Figure 5 HE-based computation outsourcing

The data owner generates the necessary keys and parameters. More precisely as secret key is needed for the description of the ciphertexts, a public key is used for the encryption and several evaluation keys for the management of the HE operations. These keys are needed for noise reduction and ciphertext re-sizing and they are used only by the Cloud Service Provider (CSP).

The data owner encrypts all data and sends the ciphertexts to the CSP. Then, the CSP performs the operations using the evaluation keys. As described in the Appendix 6.1, some operations are very expensive to be performed by the CSP. Thus, we have decided to adapt a hybrid model in which these operations are performed on plaintext data by the client. When necessary, the CSP interacts with the client to receive the output of these operations HE encrypted.

Preconditions and Input

- The client has text files of "point clouds" and triangle mesh (three floating point numbers for each point are used). This input will be provided by ALES. In particular, the file representing the point cloud is a text file with NC rows and 3 columns, where NC is the number of points forming the point cloud. The file representing the mesh is a text file having NM rows and 9 columns, where NM is the number of triangles forming the mesh and the columns are 3*3 floating points representing the 3D coordinates of the 3 vertices of each triangle.

Postconditions and Output

- The output is one floating point for each point, representing the distance between the point and the Mesh. In particular, it is the minimum of all the point/triangles distances.

Component Design

The component is written in C++ and builds on the MS SEAL library. Following the analysis that appears in the Appendix, the CKKS HE scheme has been selected to implement the basic HE operations. The CKKS scheme operates nicely to arithmetic circuits on complex and real approximate numbers.

The HE component consists mainly of the following modules:

1. ***Setup:*** this module receives as input the desired characteristics, like the security parameter λ and the input number scale and produces the various parameters *Params* of the scheme, defining mainly the size of the ciphertext, the noise and the key spaces. Also, it runs the key generation algorithm that generates the secret key, the public keys and the evaluation keys.
2. **Encryption/Decryption**: this module encrypts a plaintext and decrypts a ciphertext. For the encryption it uses as input the public key and for the decryption it needs the secret key. In both cases, the HE parameters Params are also input.
3. **CSP computations**: the CSP performs only HE additions and multiplications. For the noise and ciphertext size management, after each multiplication, the refresh and re-linearization operations are also performed using the evaluation keys. This module is not general. It is designed tailored to the needs of our algorithm.
4. **Client computations**: the client performs only number comparisons, divisions and computes the square root function. All these operations are applied on unencrypted data and they can be implemented very efficiently on the client's machine. However, they are very expensive for HE that it is implemented on p-ary circuits. This module is not general. It is designed tailored to the needs of our algorithm.

The HE component will have a few more auxiliary modules.

1. **Client/CSP communication**: this module is responsible for the interaction between client and the CSP during the computations.
2. **HE assessment**: this module evaluates the performance of the component. It mainly measures the accuracy of the computations.
3. **TEE computations**: we will investigate a trade-off between communication overhead, computation cost and security. More precisely, the intermediate operations performed by the

client will be run locally by a TEE. The TEE will have the secret and the public key and it will perform all the operations on unencrypted data.

Addressed COLLABS Common Security Requirements

Using the requirements definitions as identified in D1.2, the integration of the HE component in the COLLABS' use cases will address the following requirement:

- CSR_08: Confidentiality.
  - FHE encryption protects confidentiality of data in transit and at rest is supported. In particular, confidentiality of data transferred to and managed in the Cloud should be protected. As FHE schemes are based on the Learning with Errors problem (LWE), they also offer postquantum protection.

Beyond the state of art

The state-of-the-art security model for cloud based computation assumes that the cloud service provider is part of the trusted base. This is a very weak assumption that strongly depends on the type of data. A risk analysis will exclude the adaption of the cloud paradigm when the data at stake are critical for the data owner (financial data, health). In COLLABS, we solve this problem by using homomorphic encryption, a very powerful cryptographic tool.

The HE schemes are still rapidly evolving and a lot of problems are still not solved. In COLLABS we aim to advance the knowledge in the field and solve some of these problems. HE is notoriously weak to deal with non-linear functions, like square root functions, and with data flow controls like comparisons and loops. However, all these computations are very common and they appear in our use case algorithm. We introduce new practical approaches for loops and comparison computation by presenting a trade-off between security and communication overhead. We implement and evaluate all our solutions. In COLLABS, we are using the state-of-the-art HE schemes implementations, like the MS SEAL library[7].

Component Evolution

This is the first version of the component. We used the first year to identify the use case and the most adequate HE schemes, as well as the optimal parameters, in terms of security and efficiency. The whole procedure was performed manually by the COLLABS team of experts. The analysis of our design philosophy and our approach in general, appears in the Appendix 6.1 of this document.

The identified use case is part of the ALES scenario S2. We have implemented HE version of an algorithm the computes Euclidean-like distances to compare products with the corresponding prototypes using 3D-model representation. It is written in C++ and it is using the CKKS HE scheme as it is implemented in the Microsoft SEAL library.

At this stage, the component's implementation will serve as a proof-of-concept for our design.

Outlook

HE encryption component will be used for securely outsourcing computations to the cloud as part of the ALES scenario S2 "Controlled share of compliance data" (see D1.3). At month 18, the proof of concept is demonstrated.

[7] https://github.com/microsoft/SEAL/tree/main/native/src/seal

In the next iteration, the design will evolve in two directions. We will revisit the security model and we will introduce hardware security in the trusted base by using the INTEL SGX technology. At the same time, we will improve the implemented algorithm to optimise parallel execution in order to evaluate more accurately the practically of the proposed solution.

3.3 Workflow-Driven Security for Supply Chain and Compliance in Manufacturing

3.3.1 Workflow-Driven Security Framework (WDSF)

The workflow-driven security framework (abbrev. WDSF) *(Kasinathan & Cuellar, 2019)* is used to model, implement, and enforce complex cross-organizational business processes by making use of a distributed ledger architecture. WDSF can enable workflow-aware access control across organizational boundaries and allows digitising distributed business processes and enforcing their compliant execution. In COLLABS, WDSF is used to model collaborative manufacturing and remote maintenance use cases. In such scenarios, stakeholders of different organizations – such as manufacturers, suppliers, or service providers – collaborate and, thus, must establish and manage trust across organizations. WDSF addresses these requirements by providing transparency and accountability (non-repudiation) of workflow step execution through a shared (distributed) immutable ledger.

Functional Description

WDSF is a security framework that is used to:

- Model, specify, simulate, validate, and verify the digital representation of business processes (i.e., modelled as one or more workflows).
- Enforce correct executions of distributed business processes, denoting the adherence to the defined steps of processes and compliance with pre- and post-conditions.
- Guarantee workflow integrity, traceability, and accountability of actions.

WDSF includes a **manual** modelling phase and a **semi-automated** deployment and execution phase as illustrated in *Figure 6*, below. First, via a sequence of 4 interactive modelling steps, the workflow is designed and implemented using Petri Nets – these 4 steps are done manually:

(1) In the first step, the **business process** needs to be defined and the interaction between the stakeholders needs to be clarified. In the example of a remote maintenance scenario, this phase involves representatives of the manufacturing company (such as shop floor manager and enterprise IT manger) as well as service providers (e.g., key account manager of the maintenance company).
(2) Based on the initial stakeholder's input, the process activity is modelled via user friendly **activity diagrams**. Usually, one of the stakeholders involved creates the activity diagram.
(3) The activity diagrams get **reviewed** by all the stakeholders involved in the first step and approved.
(4) Next, a Petri Nets based **workflow** is modelled representing the business process activity diagram. This step also includes simulation, validation, and verification of Petri Nets workflow properties such as deadlock freeness and soundness. After successful validation, the Petri Nets can be directly executed via the workflow application.

The deployment and workflow execution are shown in the second half of Figure 5 where the workflow gets implemented via the following steps.

(5) Based on the Petri Nets workflow, **smart contracts**[8] are created. Thereby, the basic structure of smart contracts can be automatically generated *(Zupan, 2020)*. This approach reduces both the effort for smart contract development and interpretation errors while implementing the business process specified in the first step. Specific semantics, like the evaluation of preconditions represented in custom data structures are use case specific and cannot be automatically generated so that this step is performed semi-automatically. Note: The Petri Nets workflow may contain additional workflow steps that need not be represented in the deployment of the smart contracts. This feature is used to preserve confidentiality aspects of individual organizational workflows that need not be shared with other parties.

(6) In the next step, the smart contracts are deployed on the distributed ledger, e.g., in Hyperledger Fabric each organization (stakeholders) should accept the smart contract before it becomes functional. The architectural layers, including the blockchain, are described in more detail in the next sub-section.

(7) Finally, the workflow can be executed. Based on the Petri Nets based workflow model and implementation, compliance with the business process is enforced. Parallel workflow execution is supported via the eventual consistency provided by the underlying blockchain for synchronizing the workflow events across multiple actors in different organizations.

Steps (5) to (7) are executed semi-automatically using WDSF. Please note that all steps are shown in consecutive order. In case of changes in the business process or if any rectifications are needed, previous steps will be re-executed. These iterations have been omitted in the figure for the sake of brevity.

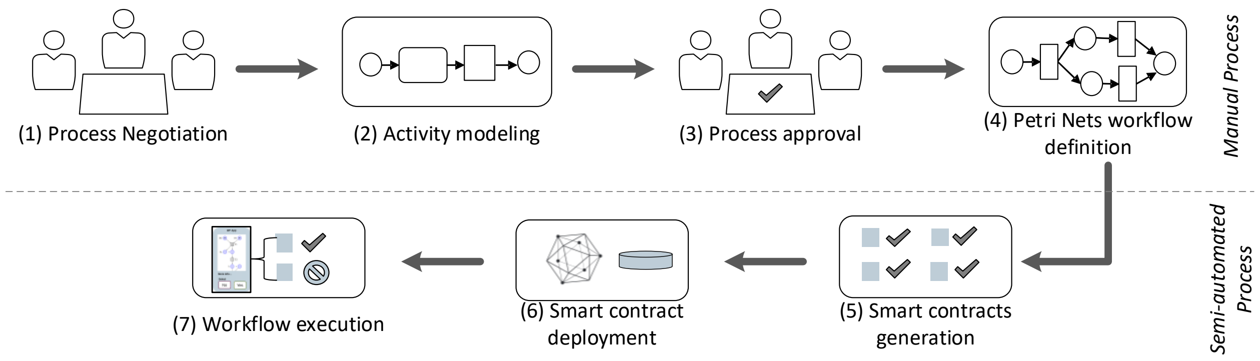


Figure 6 WDSF process steps.

After having performed the steps as illustrated in *Figure 6*, a digital implementation of a business process is provided. The business process' workflow is specified and implemented via a Petri Nets. WDSF expects workflow definitions using PNML (Petri Nets Markup Language, which is specified via ISO/IEC15909-2[9]) as standardized import/export format. The business logic – i.e., the validation of preconditions and the implementation of postconditions of a step in the process – is realized via the Petri Nets workflow engine embedded within the workflow application. The Petri nets workflow engine may interact with other services via REST APIs such as an IIoT device for triggering actions on the device or smart contracts when a workflow step's action and the corresponding information (i.e., input and output data that needs to be maintained as part of an audit log) gets stored in the underlying blockchain for auditability purposes.

[8] *Smart contracts are executable programs. These are used to control, document or automatically execute actions (can also be legally relevant events), according to the terms of a contract that stakeholders have agreed upon. See, for instance, https://en.wikipedia.org/wiki/Smart_contract and https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html#*
[9] *https://www.iso.org/obp/ui/#iso:std:iso-iec:15909:-2:ed-1:v1:en*

The blockchain can be set up as a distributed network with different stakeholders (such as manufacturers and maintenance companies) having access to it.

Preconditions and Input

WDSF exposes a Web application interface (GUI) for workflow execution. For a workflow to be executed via the Petri Nets workflow engine, WDSF uses an upload interface. Workflow steps and their validations are use-case specific. Hence, input parameters and metadata must be evaluated as preconditions of transitions. In the context of COLLABS these input parameters could be access tokens (e.g., JSON Web Tokens, JWTs) representing access permissions to perform a certain step in the process. Input tokens can be evaluated by two different components in the WDSF architecture:

a. via the Petri Nets workflow engine;
b. via smart contracts deployed in the blockchain.

Depending on the use case specific conditions, validation can be done by either one or by both components.

Postconditions and Output

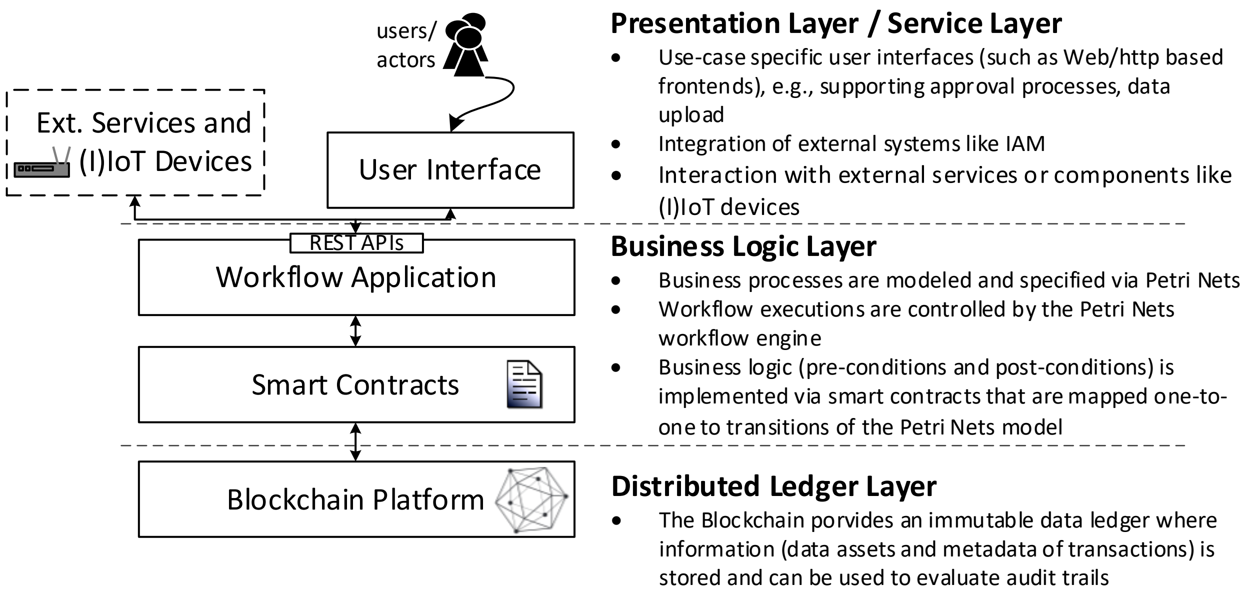
The Petri Nets-based workflow engine transforms states of the workflow based on input provided by actors into new states with workflow-specific post-conditions. That is, after executing a transition in the workflow, well-defined states in the Petri Nets are reached. In addition, the transition can also generate tokens as output that represent postconditions. WDSF uses REST APIs for interacting with external applications, such as IIoT devices or IoT services and Web-based applications. In the context of COLLABS, output tokens can be simple assertions of a state, or access tokens to access next resources (e.g., JSON Web Tokens, JWTs). Generation of output tokens can be implemented in two different components of the WDSF architecture:

a. as part of the Petri Nets workflow engine;
b. via an external authorization or IAM component deployed together with WDSF.

It depends on the trust of the workflow execution application. That is, if the workflow execution engine and its application are trusted and they run in a tamper-proof environment, then the workflow execution application can itself generate access tokens; otherwise, it is recommended to use a state-of-the-art authorization server to generate access tokens. In this case the workflow application needs to establish a secure connection to the authorization server.

Component Design

Figure 7 provides a high-level overview of the system architecture of the Workflow-driven Security Framework (WDSF).



Presentation Layer / Service Layer
- Use-case specific user interfaces (such as Web/http based frontends), e.g., supporting approval processes, data upload
- Integration of external systems like IAM
- Interaction with external services or components like (I)IoT devices

Business Logic Layer
- Business processes are modeled and specified via Petri Nets
- Workflow executions are controlled by the Petri Nets workflow engine
- Business logic (pre-conditions and post-conditions) is implemented via smart contracts that are mapped one-to-one to transitions of the Petri Nets model

Distributed Ledger Layer
- The Blockchain porvides an immutable data ledger where information (data assets and metadata of transactions) is stored and can be used to evaluate audit trails

Figure 7 WDSF system architecture.

The system components are:

User Interface (UI)	We assume that web portals (e.g., for an approval process in the context of a remote maintenance access process) are provided for human users to interact with the business process, e.g., to pick up tasks and approve certain steps. The UI is designed to be user friendly and responsive to different media devices by presenting necessary information for novice users and advanced options are hidden by default. The users can visually validate the workflow state, query workflow history, and find which places and transitions they are authorized to interact. Web UI developments that are addressing specific use case requirements are not in scope of the WDSF development. Instead (e.g., in the context of ALES use case Scenario 1), we are focussing on providing a UI to illustrate the interaction of the workflow execution layer with the underlying distributed ledger layer.
Workflow Execution Application	The core-components of workflow execution that is implementing and enforcing the application logic are as follows: Petri Nets Execution Engine, Workflow APIs designed and developed for specific use cases, and authorization components developed for validating Petri Nets tokens or access tokens. The Petri Nets specification and the conditions written are enforced in this layer. (Petri Nets-based) WDSF workflow application is thus enforcing the workflow execution and workflow compliance.
Smart Contracts	Smart contracts represent the second layer in the WDSF architecture where the application logic is implemented. Smart contracts for instance perform semantics checks, like validation of tokens and signatures to check that the approval of a request has been performed by an authorised user. Furthermore, the smart contracts layer also acts as a policy enforcement point (PEP), evaluating whether a requestor/user can perform a certain step

	in the workflow. Such checks are implemented via so-called smart contracts on top of the blockchain layer.
Blockchain Platform	A blockchain infrastructure is used to store any transaction, i.e., any activity in the workflow. The benefits of using blockchain technology are: • **Immutability** of information stored in the blockchain. That is, the blockchain infrastructure is used as a reliable, distributed audit log that ensures non-repudiation of actions. • **Authenticity** of actions, meaning that the originator of an action can be identified and authenticated. Please note: the main requirement is to be able to identify the origin of actions. That applies to many industrial use cases. Therefore, anonymity – which represents another key feature of a blockchain infrastructure – is not used, by intent. • **Decentralization** denotes the possibility to distribute information across the nodes of different stakeholders. This depends on the deployment scheme used for the blockchain infrastructure. • **Transparency**, meaning the possibility to distribute information across participating nodes so that all authorised parties can get access to shared information.

Via the API provided (e.g., in form of REST interfaces) by the Workflow Execution Application, also external services and components can interact with WDSF. For instance, in the context of ALES use case scenario S1, the control plane of the SDN (software defined network) of the ALES laboratory will query access control status information from WDSF for configuring network access for external users.

The overall system architecture (as illustrated in Figure 7) follows a three-tier architecture design with the **UI** being realised in form of web portals, the **application layer** being realised through (a) the Petri Nets-backed WDSF workflow application and (b) smart contracts, and the **data layer** in form of the distributed ledger/blockchain.

The benefit of separating the application layer into (a) and (b) is that we can use the best out of both technology stacks, meaning that the Petri Nets-based WDSF workflow application focuses on enforcing the workflow/business process (i.e., execution paths) while additional conditions are checked (=pre-conditions) and created (=post-conditions) by smart contracts that have direct access to information stored in the blockchain or off-chain.

The implementation of WDSF is based on the following technology stacks/3rd party libraries:

- For the **web portals** (for demonstration purposes) we make use of use case specific web applications that were developed using the **FLASK**[10] framework.
- The **WDSF** workflow engine is realized using **SNAKES**[11] which is a Python library used to define and execute Petri Nets-based workflows.
- As underlying distributed ledger architecture, we make use of **Hyperledger Fabric**[12] with smart contracts implemented in **JavaScript**.

[10] *FLASK - https://palletsprojects.com/p/flask/*
[11] *SNAKES - https://snakes.ibisc.univ-evry.fr/*
[12] *Hyperledger Fabric - https://www.hyperledger.org/*

Addressed COLLABS Common Security Requirements

Via the integration of WDSF in COLLABS' use cases, the following requirements as identified in D1.2 will be addressed:

- CSR-03 Authorization (timed access)
  - WDSF will evaluate the context of requests (e.g., via checking access control tokens including expiration dates), thus granting access for the least required time, only
- CSR-04 Authorization (least privilege)
  - WDSF is used to implement the business process of remote maintenance in a controlled and restrictive way, denoting that the least-required access path for a remote service engineer will be realized, only.
- CSR-07 Logging
  - WDSF's architecture includes a distributed ledger technology (e.g., Hyperledger Fabric) which is used to log each important interaction of the workflow participant (e.g., maintenance person, devices, etc), ensuring integrity and availability of log data.
- CSR-09 Accountability and non-repudiation
  - Accountability and non-repudiation are fulfilled by WDSF through the underlying blockchain architecture as each interaction is logged by ensuring authenticity (using digital signatures)

Beyond the state of art

So far, no existing work focuses on secure remote maintenance by applying distributed trust, workflow-based access control and auditable records within an industrial use case scenario. Precisely, identifying and analysing the security requirements, improving the security by defining protection requirements, restricting and validating the access by using formal modelling techniques such as Petri Nets, and finally, improving transparency and accountability of actions performed in an industrial context.

In this section we describe three core technologies that can be used to solve the secure remote maintenance use case a) Petri Nets for modelling and enforcing workflow; b) blockchain for immutability and accountability; c) finally, commercially available Privileged Access Management (PAM). We show how our approach (WDSF) goes beyond the of state of art PAM technologies by integrating and extending the above mentioned two core technologies (a) and (b). In terms of modelling workflows, validating and verifying them, some of the most prominent approaches are the following: Petri Nets, automata, process algebra, business process modelling notation (BPMN). The advantages and the reasons for adopting Petri Nets are presented in *(Kasinathan & Cuellar, 2019)*. A review of blockchain application in next generation cybersecurity application in Industry 4.0 context is presented in *(Fernández-Caramés & Fraga-Lamas, 2019)*, and this review article shows that blockchain can enhance industrial technologies by adding decentralized security, trust, immutability, with a higher degree of automation through smart contracts. A generic blockchain enabled cyber-physical-system (CPS) architecture for Industry 4.0 manufacturing systems is presented by Lee et al., in *(Lee, Azamfar, & Singh, 2019)*. Secure industrial remote maintenance by using software defined networking (SDN) and attributed-based access control (ABAC) is presented in *(Kern & Anderl, 2019)*.

Several commercial Privileged Access Management (PAM) solutions exist that enable secure remote access, and a report on their critical capabilities is presented in *(Kelley, Gaehtgens, & Data, 2020)*. Most of these solutions consider centralized trust and control and focus on security aspects such as multi-factor authentication, just-in-time (JIT) access and zero standing privileges (ZSP) realising the principle of least privilege. However, with WDSF we focus on having distributed trust while resource providers have complete control over their resources while enforcing, compliance, traceability and auditability of business processes/workflows.

Component Evolution

Since the MVP release, the WDSF component has been steadily improved and developed further. Amongst others, the following key updates and extensions have been achieved:

- **Platform maintenance:** the release pipeline of the used third-party components is steadily monitored and updates are integrated into WDSF in short time. Earlier in this section (see above) we provide a detailed overview about the components used by WDSF. One focus was on updating the underlying blockchain infrastructure. In this regard, the long-term support version 2.2 LTS of Hyperledger Fabric[13] has been integrated into WDSF, replacing the former version 1.4. Version 2.2 is the first long-term support (LTS) release of Hyperledger Fabric v2.x and has been chosen to ensure code stability and maintainability which are considered key qualities of WDSF, as well. In the course of this upgrade, interfaces of WDSF to the underlying HLF layer got updated and tested, accordingly.
- **Updates of business logic layer:** Significant updates to the workflow application have been implemented. Amongst others, the Petri Nets workflow execution history information is now made available (see also UI improvements, below). Apart from providing the possibility to retrieve the workflow history as part of the audit log, also for cases where certain business logic steps are not modelled via smart contracts (and thus shared across cooperating partners in the blockchain network), these steps can also get logged on in the workflow log of the affected organization. This fine-grained log-separation (i.e., organization-specific log vs. cross-organizational log) allows to concisely handle confidential log information on the business logic layer. Furthermore, access control restriction on places and transitions of the Petri Nets workflows have been realized. Concerning access control for places this denotes control on who is allowed to manage (i.e., provide) input for places. Authorization rules on transitions control permissions for executing them, i.e., specifying who is allowed to fire transitions.
- **User interface improvements:** another focus of the ongoing development was on improving the user experience for WDSF. WDSF's UI is aiming at illustrating the interaction between the business process layer and the underlying distributed ledger (i.e., Hyperledger Fabric). Because of that, the nature of the UI is more oriented for expert users rather than end users such as enterprise managers. For the demonstrator (i.e., the use of WDSF in the context of scenario 1 of ALES use case, see D5.3, Section 6.1 "ALES - Use Case Scenario 1 - Controlled and secure remote maintenance"), the UI shall demonstrate user and tenant separation (i.e., access control) and audit log functionality that is ensuring non-repudiation of actions taken. These features have been further developed and integrated accordingly: For instance, user roles are now visible in the UI and only such functionality which is allowed to be executed (enforcement is implemented in the underlying middleware and blockchain layer) is enabled. Figure 8 illustrates that user experience. In addition, as shown in Figure 9 access to the audit log is included, so that authorised users can retrieve history information like preceding approval steps.

[13] https://www.hyperledger.org/blog/2020/07/20/new-release-hyperledger-fabric-2-2-lts and https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatsnew.html

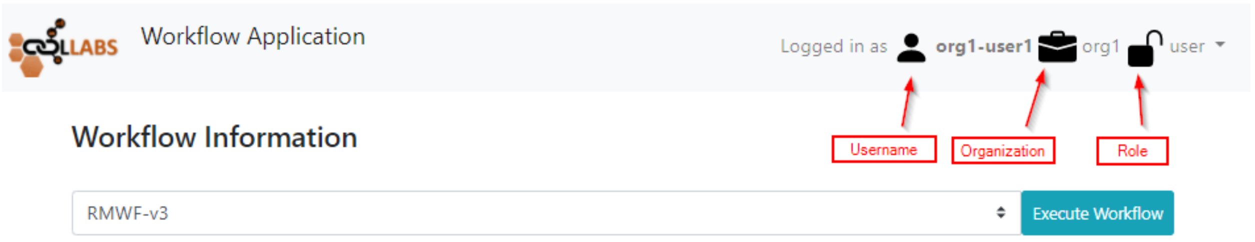
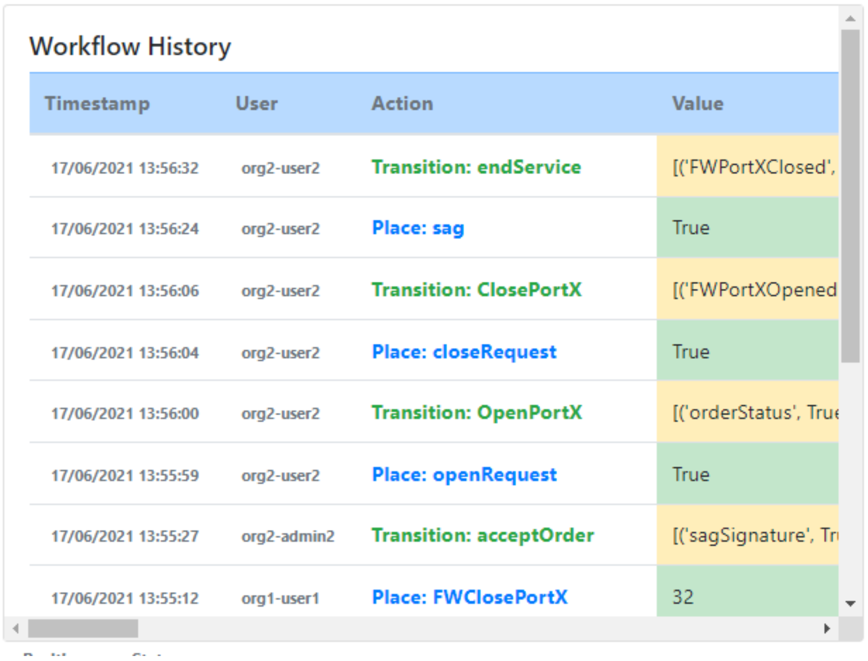


Figure 8: WDSF's Workflow-Application UI improvements showing username, organization and role of the user.



Workflow History

Timestamp	User	Action	Value
17/06/2021 13:56:32	org2-user2	Transition: endService	[('FWPortXClosed',
17/06/2021 13:56:24	org2-user2	Place: sag	True
17/06/2021 13:56:06	org2-user2	Transition: ClosePortX	[('FWPortXOpened
17/06/2021 13:56:04	org2-user2	Place: closeRequest	True
17/06/2021 13:56:00	org2-user2	Transition: OpenPortX	[('orderStatus', True
17/06/2021 13:55:59	org2-user2	Place: openRequest	True
17/06/2021 13:55:27	org2-admin2	Transition: acceptOrder	[('sagSignature', Tr
17/06/2021 13:55:12	org1-user1	Place: FWClosePortX	32

Figure 9: WDSF's Workflow-App showing workflow execution history information that includes the following: timestamp, user performing the action and the input values used to trigger the workflow action.

- **Platform and component monitoring:** to ensure the security of the underlying IT platform and for being able to analyse and identify atypical executions, WDSF has been connected to the COLLABS logging and monitoring infrastructure. In this regard an Apache Kafka client got integrated and JSON-formatted log messages are sent to 3ACEs. For instance, following CIS'[14] security guidelines, NGINX server's access log is collected and analysed using COLLABS central monitoring platform. In addition to the central monitoring approach, the monitoring of WDSF services got updated via the integration of ELASTIC Application Performance Monitoring (APM) plugins.
- **Automated Deployment:** In order to ease the deployment of the blockchain infrastructure, automated deployment scripts based on minifabric[15] have been developed. These scripts allow the seamless setup of the HLF infrastructure, e.g., in cloud-based environments like AWS or Google Cloud. They are, amongst other things, used for setting up the distributed blockchain network for the demonstrator of scenario S1 "Controlled and secure remote maintenance" of the ALES use case.

[14] *Center for Internet Security, https://www.cisecurity.org/*
[15] *https://github.com/hyperledger-labs/minifabric*

Outlook

WDSF will be used for designing and implementing distributed, collaborative workflows like the remote maintenance scenario (ALES scenario S1 "Controlled and secure remote maintenance", see D1.2). A detailed development plan has been defined for the incremental setup of a demonstrator for ALES scenario S1. At M18 an initial demonstrator setup and integration of WDSF in the ALES laboratory environment will be provided. In the next iteration, the following major developments are planned:

- Smart contracts: next to the update of the Petri Nets layer, smart contracts will be modelled and implemented to complete the implementation of the business logic for the remote maintenance scenario S1 of the ALES use case. In this regard also an appropriate data model must be designed and implemented which is used to represent approval context information. Moreover, it is also planned to apply a formal verification approach to workflow specification and thus the derived smart contracts security can be improved.
- WDSF integration with factory devices and controllers in ALES scenario S1.
- WDSF integration with log visualization component into COLLABS' monitoring framework.
- WDSF integration with COLLABS federated identity management system.

3.4 Network Traffic Monitoring

3.4.1 Distributed Anomaly Detection

Functional Description

The Distributed Anomaly Detection (DAD) system is installed in an industrial network with multiple subnetworks. In each subnetwork a local unit is installed. The system works in two phases: training and detection. During the training phase, each local model train on the local traffic of the devices in its subnet. A profile for each device type is learned. If devices of the same type exist in multiple subnetworks, the local units collaboratively share the learning information between them to achieve a more accurate model for this device type. After the training phase finishes, the DAD system switches to the detection phase. In this phase, each local unit analyses the local traffic and detects anomalies in the behaviours of the devices within its subnetwork. Anomalies are detected based on deviations from the collaboratively learned profile of each device type.

Preconditions and Input

- As mentioned in the previous section, the system runs first in the training mode. We assume that during the training there are no attacks performed on the network. This is an essential requirement to make sure that the models learn correct profiles for each device type. One option to achieve this requirement is to use manually analysed data for the training. During the detection phase, each local unit should have access to the traffic passing in the local gateway (eg., using a span port). Additionally, the local units should be able to communicate with each other over the network.
- To demonstrate our tool, we use a public dataset[16] of benign IoT traffic to train the local units. Additionally, we use a test dataset with benign and malicious traffic to measure the performance of the DAD. The datasets are composed of traffic from 10 devices. Table 1 shows the number of

[16] IoT Security - IoT Traffic Analysis (unsw.edu.au)

packets used in the training and testing, and the number of malicious packets in the testing dataset. The attacks performed are of two types: direct and reflection. Direct attacks include ARP spoofing, TCP SYN flooding, Fraggle (UDP flooding), and Ping of Death. Reflective attacks include SNMP, SSDP, TCP SYN, and Smurf. Attacks are performed with different data rates.

Table 1 Statistics of the dataset used for training

Device	Training Packets	Testing Packets	Malicious Packets
WeMo motion	14,868,367	7,721,200	841,631
WeMo switch	532,953	646,662	271,488
Samsung smartcam	2,202,479	1,433,943	417,167
Tp-Link Smart Plug	55,687	367,796	273,914
Netatmo camera	1,006,719	728,121	265,031
Chromecast Altra	32,763,645	2,182,033	768,988
Amazon Echo	914,080	291,004	23,522
iHome Smart Plug	160,794	70,302	754
LiFX Bulb	394,285	175,385	63,272
Total	100,126,044	44,109,514	2,925,592

Postconditions and Output

After all the local models are trained, the DAD system enters in the detection phase. The local units analyse each packet passing through the gateway and results in a decision for each packet. The decision can also concern a set of packets belonging to the same connection, or a set of consecutive packets. The decision is a binary classification of the packet whether it belongs to an attack or not. Each packet is then labelled either as 'benign' or 'malicious'. We assume another entity to handle these decisions and take actions accordingly.

Component Design

The design of the DAD system is split into two parts: The local detection unit and the distributed learning part. We rely in the design of the local detection unit on a Mixture of Experts approach. Each local unit is composed of multiple experts (i.e., machine learning models) that specialize in detecting certain type of attacks. Three types of experts are specified according to the type of features extracted from the packet. Experts in statistical features, experts in packets metadata, and experts in packet's payloads. Each local unit has a pre-processor that extracts useful features from the analysed packet. Then, a per device-type expert is trained for each type of features. We first describe the type of features extracted and then describe the experts.

Feature types:

- Statistical Features: These features are extracted from a window of consecutive packets. The window size is a hyper parameter for the model. For each window of N packets, the following metrics are computed:
    - Traffic flow rate
    - Packet average size
    - Packet size standard deviation
    - Rate of TCP packets
    - Rate of UDP packets
- Metadata Features: These features are extracted from the headers of the packets. It includes:

  - Source Port
  - Destination Port
  - Packet Length
  - Protocol
  - TCP flags
- Payload Features: The features are the application layer's data of the packet. These are application specific.

Expert Types:

- Statistical Experts: Experts in statistical features specialize in detecting attacks that run for a significant period. These can be DoS attacks, brute forcing, network scanning…
- Metadata Experts: Experts in packet's metadata detects anomalies in the control flow. These detect more sophisticated attacks mainly TCP attacks (eg., TCP RST attacks, TCP SYN floods…)
- Payload Experts: Experts in the payloads train models specific for each application. These models learn patterns in the data transmission from a device type. This allows these experts to detect application layer attacks such as exploits on the DNS or HTTP protocols.

Additionally, each local unit runs a distribution agent that is responsible for sharing the training information. This distribution agent communicates with other agents and synchronizes with them. The goal is to train a global model per device type using all traffic from different local units. The distributed training is based on a federated learning approach. On each training step, the updates of the expert models are not directly applied. Instead, they are shared with the other local units which are training the same model. Specifically, we use a publish-subscribe paradigm to aggregate the updates. The first expert finishes a training step acts as a server and informs the others of this model update. The other local units which are training the same model subscribes to this update. When each of them finishes its training step, it sends its update to the server which aggregates all the updates and publishes the aggregated model update to all the subscribers.

Example: Assume we have 10 sub-networks containing the same type of PLC. Each local unit is training a statistical expert for its local PLCs. We use an auto-encoder model as a statistical expert. Assume the training batch size is 64. After parsing the first 64 features from the real-time traffic generated by the PLCs, the local expert starts its first step of training. We use Stochastic Gradient Descent (SGD) as the learning function. After applying SGD, the local unit generates the first model update. It informs all the 9 other local units about this update and waits for them to send their updates. When the other local units parse 64 statistical features, they apply SGD and send the model update to the first local unit. After all of them send their updates, the first local unit sums these updates and divide by 10 and send back the aggregated updates. This way all the models obtain the same global model for their statistical expert.

One potential work for this project is to integrate privacy techniques on the aggregation of the model updates. This may involve differential privacy and secure aggregation. These techniques help preserving the privacy of the data used for the local training. The drawback of adding differential privacy is degrading the accuracy. Additionally, applying secure aggregation increases the communication overhead. It depends on the scenario of deployment and requirements to decide if these techniques are needed.

Addressed COLLABS Common Security Requirements

Using the requirements definitions from D1.2, the integration of the DAD component in the COLLABS use cases addresses the following common security requirement:

- CSR-08: Monitoring
  - CSR-08 is partially covered as the DAD provides monitoring and network traffic analysis capabilities, allowing detecting anomalous behaviour and thus security related events.

Beyond the state of art

Our study focuses on the problem of anomaly detection by means of Mixture of Experts (MoE) *(Yuksel, Wilson, & Gader, 2012)*. Mixture of Experts is one of the most interesting combining methods, which has a great potential to improve performance in machine learning. This technique is established based on the divide-and-conquer principle where the problem space is divided between few neural network experts, supervised by a gating network. Most of the existing anomaly detection methods use a single low-dimensional manifold that might not be ideal for the classification of anomalies. However, modelling each detection unit composed of multiple experts that are able to detect certain type of attacks, is promising to classify anomalies. In our study, three types of experts are specified according to the type of features extracted from the packets: experts in statistical features, experts in packets metadata, and experts in packet's payloads. To the best of our knowledge, this study is the first to provide distributed anomaly detection by means of MoE.

In *(Nguyen, et al., 2019)*, the authors propose a framework DIoT that is a self-learning system for detecting compromised devices. It deploys device-type specific anomaly detection. DIoT applies a federated learning approach for aggregating the anomaly detection profiles for the intrusion detection. Our proposed system does not only leverage device-type specific anomaly detection, but it also provides more accurate view about the overall network behaviour to be able to detect attacks at different levels. Compared to the DIoT, we leverage the use of Mixture of Experts in order to detect certain type of attacks, and then classify anomalies.

Component Evolution

At this point, we have tested 3 possible models for the statistical experts and 3 metadata experts. We trained each model on the training dataset containing only benign data, and evaluated it on the test dataset contained a mixture of benign and malicious data. To evaluate the performance of each model, we compute the F1-Score of each classifier. For each device type we create a statistical expert. The evaluation of each type of statistical expert for each device type is demonstrated in the following:

Statistical Experts:

- Local Outlier Factor: Is a machine learning model based on computing the local density of each input. This model supports unsupervised learning and is used for novelty detection.
  - F1-Score = 0.913
- Isolation Forest: This model uses decision trees to classify the data. In order to isolate a data point, the algorithm recursively generates partitions on the sample by selecting an attribute and then selecting a split value for the attribute, between the minimum and maximum values allowed for that attribute. The data points that can be easily separated are considered anomalies.
  - F1-Score = 0.903
- Auto-Encoders: Auto-Encoders are Artificial Neural Networks aiming at reconstructing the inputs they receive. Auto-Encoders are efficient at reconstructing inputs data similar to their training data. On the other hand, they have large errors trying to reconstruct inputs data different of the training data. These properties make them suitable to detect anomalies related to a training dataset. The Autoencoder accepts high-dimensional input data, compress it down to the latent-space representation in the bottleneck hidden layer; the Decoder takes the latent representation of the data as an input to reconstruct the original input data. During the training, we try to

minimize the reconstruction error. Later during detection phase, anomalies are detected by checking the magnitude of the reconstruction loss.
  - F1-Score: 0.919
- Auto-Encoders w/t LOF: To optimize the detection model we create an ensemble classifier based on the trained Auto-Encoder and the LOF models. The classifier measures the confidence of each model on each input and gives the result of the highest confidence model. The confidence metric is computed in the LOF factor based on the normalized distance of the input to the already seen benign inputs. Similarly, the confidence of the auto-encoder is measured based on the distance of the construction error to the pre-set threshold:
  - F1-Score: 0.94

Metadata Experts:

- Recurrent Neural Networks (RNN): RNNs are neural networks designed with an internal memory. Using a feedback loop on the internal neurons, the RNN can remember important things about the input they receive. This makes them perfect for sequential data such as the metadata features we extract from the packets.
  - F1-Score: 0.86
- Long-Short-Term Memory (LSTM): LSTM are special types of RNNs. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.
  - F1-Score: 0.9193
- Gated Recurrent Units (GRU): Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks. The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate.
  - F1-Score: 0.9197

In summary, according the evaluation results, we choose to use the ensembled classifier (i.e., Auto-encoder w/t LOF) as the statistical expert and the GRU model as the metadata expert.

Local units are deployed in Docker[17] containers. Docker containers allow lightweight virtualization, which however uses some features of the host operating system. A publish-subscribe communication broker has also been set up. As a first step, an Apache Kafka[18] communication broker is set up, in order to provide a communication solution to the different local units. The purpose of this communication broker is to manage the communications concerning the variables and optimized parameters of the machine learning models used by the local units. Topics are generated for each local unit. On these topics, the corresponding local unit publishes the parameterizations of its machine learning models. Other local units subscribe to local unit topics relevant to their models and can update their settings based on models trained on different sets of relevant data. At the local unit level, Federated Learning algorithms or aggregation of machine learning models can be implemented.

[17] *https://www.docker.com/*
[18] *https://kafka.apache.org/*

As a second step, the study of a lighter communication broker such as mosquitto MQTT[19] is underway. In the event that the risk of having a Single Point of Failure at the level of the communication broker is too important, point-to-point communications are studied for the exchange of cyber threat intelligence or machine learning information between the local units.

Local units and the communication broker are managed with Docker-compose[20], a tool for orchestrating multiple Docker containers.

Outlook

Various areas for improvement are underway or under study for the Distributed Anomaly Detection component.

Training on the specific data of use cases is a very important point. It allows us to have the most efficient models possible in relation to the monitored infrastructures.

The precise determination of the types of attacks taking place in the monitored network is also of great interest. Further study of packets classified as abnormal by more specific attack detection models or signature approaches is a possibility. It helps reduce the rate of false positives. In addition, the possibility of generating Cyber Threat Intelligence reports in a standard format such as STIX is under study. This allows the local units to make standardized reporting, usable by the other local units of the DAD, internal components of the COLLABS framework, as well as external components.

One possible barrier to the adoption of Distributed Anomaly Detection using Federated Learning is the possible information leakage on training data. It is possible to infer training data, using differential privacy attacks on models computed by Federated Learning. In cases where training data is sensitive, has high value or has confidentiality constraints, the use of Federated Learning by traditional approaches may be problematic. For the Distributed Anomaly Detection component, network traces are used for training. In certain deployment cases where the local data units are deployed at different tenants, these network traces can contain sensitive information on the network topology, the attacks that have already targeted the infrastructure, etc. We are currently studying the use of procedures of Secure Computing (Secure Multi Party Computation and Homomorphic Encryption) in order to address this privacy issue about training data. The goal being to offer models for detecting anomalies trained by Federated Learning and robust against attacks by Differential Privacy. This allows models to be trained on the union of local units' training data, while respecting the confidentiality of local private data.

3.4.2 Encrypted Traffic Analysis

Functional Description

The encrypted traffic analysis module can detect malicious network traffic using signatures which have been created based on patterns of packet payload lengths. The signatures describe packet sequences for the identification of intrusion attempt events in encrypted networks using packet metadata. For the generation of these signatures, we use publicly available labelled datasets. We process the traffic collected from these datasets and only keep traffic relevant to intrusion events. Then we break it into flows and extract the sequences of packet payload sizes per flow. Finally, we use the VMSP algorithm

[19] https://mosquitto.org/
[20] https://docs.docker.com/compose/

(Fournier-Viger, Wu, Gomariz, & Tseng, 2014) for frequent sequential pattern mining to generate the signatures based on the sequences of packet payload sizes.

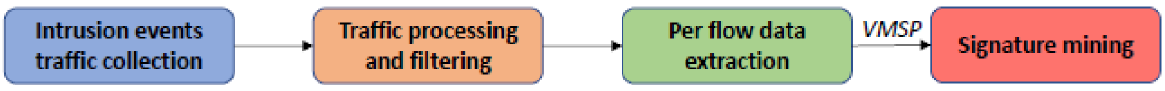

Intrusion events traffic collection
Traffic processing and filtering
Per flow data extraction
VMSP
Signature mining

Figure 10 Illustration of methodology workflow

Preconditions and Input

- Dataset of network traffic which contains labelled malicious flows.
- Generated database of signatures.
- Access to traffic generator which can produce streams of malicious and benign traffic to look for intrusion attempt events.

Postconditions and Output

- Output an alert if there is a match by breaking the network traffic into flows and searching for these specific signatures.

Component Design

To enhance our database of malicious threats in encrypted traffic we used a publicly available dataset to produce signatures. The IoT-23 Dataset *(Parmisano, Garcia, & Erquiaga)* is a group of labelled datasets which contains flows in both unencrypted and encrypted traffic from malware which are known for recent attacks in IoT environments like the mustik[21]. Each dataset contains both benign and malicious traffic for a specific malware. The label of the malicious traffic describes a specific action taken by the malware such as 'C&C' or 'attack'. 'C&C' action means that the malware established a connection and transferred information between itself and a C&C server while the 'attack' means that some kind of attack was executed. Other labels exist such as file download and network scans.

To produce malicious signatures of malware actions based on packet sequences and payload size we carefully analyse each dataset containing malware traffic. We fragment all of the network traffic into flows and then group the flows based on their labels. Since our mechanism relies on packets with payload, we only keep flows which have at least 1 packet with payload. An example of the flows we group together can be seen in the figure below where we can see 6 flows from the 828 flows in total of the IRCbot malware with the label 'attack' on the SSH port.

```
192.168.1.194 X.X.131.66 60868 22 6 1547066139.530965 42 42 42 42 42 42 42 42
192.168.1.194 Y.Y.165.41 35398 22 6 1547066109.316691 42 1432
192.168.1.194 Z.Z.233.109 42266 22 6 1547066181.280917 42 1432 48 16 44 60 84
192.168.1.194 E.E.173.16 58442 22 6 1547066615.335017 42 1432 48 16 44 60
192.168.1.194 F.F.188.87 34464 22 6 1547066796.856312 42 1432 48 16 44 60 140
192.168.1.194 W.W.106.183 60528 22 6 1547066836.990442 42 1432 48 16 44 60 84
```

Figure 11 Example of grouped flows for specific label

[21] *https://unit42.paloaltonetworks.com/muhstik-botnet-attacks-tomato-routers-to-harvest-new-iot-devices/*

In the figure the first two columns represent the source IP address which belongs to the malware and destination IP addresses which have been anonymized. Then we have the source port, destination port, type of IP packet and the timestamp. After the timestamp each number represents the payload length of each packet which belongs to that flow.

To allow for fast signature generation we automate the procedure as much as possible. We extract the intrusion signatures from network packet traces using a frequent sequential pattern mining technique. From our sample collection, we detect frequent packet payload size sequences that correspond to specific intrusion attempts. Frequent sequential pattern mining techniques are used to discover frequent sequential patterns that occur in sequence databases. Benefiting from such techniques, in our methodology we choose to utilize a maximal sequential pattern mining algorithm. The maximal sequential pattern mining technique is used to extract the frequent longest common sequences of network packet payload sizes contained in the traffic.

After gathering a list of patterns for a specific event we inspect the packets which belong to that pattern. We discover that in some cases as depicted in the first flow of the figure above it was retransmission of a single packet due to possible network issues. In some other cases as seen in the second flow, the communication between the malware and the client was interrupted or not complete. After removing these erroneous patterns from our collection, we were left with the signature '42 1432 48 16 44 60' which describes a secure key exchange of the SSH protocol and some encrypted attack packets from the malicious host. We follow a similar approach for other malware and specific labelled actions and generated unique signatures which correlate to completed malicious actions and can be seen on the table below.

Table 2 Evaluation of generated signatures

Malware	Label	Signature	Hosts with label	Hosts with signature	Detection rate	Hosts with erroneous action	Detection rate without erroneous actions:
Hide&Seek	*Network scan*	8 8 7 4	54	28	52%	24	93%
Hajime	*Network scan*	3 9 3 12 3	31	18	58%	10	86%
IRCbot	*C&C*	16 23 19 13	5	5	100%		
IRCbot	*Attack*	42 1432 48 16 44 60	345	172	50%	66	62%
Muhstik	*C&C*	78 15 31 47	3	3	100%		
Muhstik	*Attack*	21 152 144 16 52 68 84 52	189	118	62%	71	100%

In *Table 2* we can see the malware and the type of the action we created the signature for. For evaluation purposes we collected all the hosts which had flows with the label we constructed for each specific signature. We kept all of the flows which had at least 1 packet with payload. We then compared the pool of labelled hosts with the hosts which contained the signature we produced to calculate the detection rate of our mechanism.

As it can be observed in the table the detection rate of certain actions is low in comparison to others, so we decided to investigate further. It was clear that for the majority of the hosts where we did not detect a match it was due to the action not being complete due to a premature ending or fragmentation from retransmissions. We decided to measure these hosts which contained an incomplete version of our signature or a fragmented version for that action to verify the accuracy of our signature. To do that we measured all of the hosts which flows ended prematurely and their length was smaller than the signature itself. We also added to that group the hosts which contained retransmission in their flows. We named this group 'Hosts with erroneous action' and as it can be seen on the table if we do not take them into consideration the accuracy of our signature rating improves drastically.

Additionally, we wanted to make sure our approach will not result in high false positives. To calculate our false positives, we tested our signatures against the benign traffic included in the dataset and had zero matches. The benign traffic can be found as "Normal Captures", part of the "Malware Capture Facility Project". The signatures generated by our methodology were searched against the traffic captures that contain HTTPS network traffic.

Finally, CISCO's Joy[22] is used to provide extra functionality to our traffic analysis module. By capturing TLS handshakes, it can extract information from packet metadata and produce information on the versions of the TLS used. We use this information to produce alerts if we find a deprecated version of TLS used which can lead to privacy leaks.

Addressed COLLABS Common Security Requirements

- *CSR-08 (monitoring) The component partially covers this requirement by identifying security events in encrypted traffic and producing alerts.*
- *CSR-18 (monitoring) The component partially covers this requirement by utilizing Cisco's JOY module which classifies and reports on the version and strength of the crypto used.*

Beyond the state of art

Popular network intrusion detection systems (NIDS) like Snort[23] and Suricata[24] utilize pattern matching and regular expressions matching algorithms in order to analyse network traffic. However, the majority of these works are based on methods that extract content from network packet payloads to match suspicious signatures. Traditional deep packet inspection is becoming insufficient for encrypted network traffic (e.g., SSL/TLS protocols).

BlindBox *(Sherry, 2015)* performs deep-packet inspection directly on the encrypted traffic, utilizing a new protocol and new encryption schemes. *(Shone, 2018)* propose a system that combines deep learning techniques to provide intrusion detection. *(Tang, 2016)* present a deep learning approach for flow-based anomaly detection in SDN environments, while *(Niyaz, 2016)* utilize deep learning in order to detect DDoS attacks in such environments. *(Anderson, 2017)* compare the properties of six different machine learning

[22] *https://developer.cisco.com/codeexchange/github/repo/cisco/joy/*
[23] https://www.snort.org
[24] https://suricata-ids.org

algorithms for encrypted malware traffic classification. Moreover, *(Amoli, 2016)* present a real-time unsupervised NIDS, able to detect new and complex attacks within encrypted and plaintext communications. Kitsune is a NIDS, based on neural networks, and designed for the detection of abnormal patterns in network traffic *(Mirsky, 2018)*. It monitors the statistical patterns of recent network traffic and detects anomalous patterns. Moreover, *(Nicolás Rosner, 2019)* presents a black-box approach for detecting and quantifying side-channel information leaks in TLS-encrypted network traffic. These techniques identify malicious events in the network, by examining the characteristics of the underlying traffic, using exclusively machine learning approaches. Many research and commercial solutions focus on inspection of encrypted network traffic mostly for network analytics *(Conti, 2016)*, *(Lotfollahi, 2019)*, *(Taylor, 2017)*. OTTer *(Papadogiannaki, 2018)* is a scalable engine that identifies fine-grained user actions in OTT mobile applications even in encrypted network traffic. *(Orsolic, 2016)* use machine learning for the estimation of YouTube Quality of Experience. To test their approach, authors collect more than 1k different YouTube video traces under different bandwidth scenarios. *(Mazha, 2018)* investigate the Quality of Service of video in HTTPS and QUIC protocols. The set of features that expose usable information is based on (i) network and transport layer header information for TCP flows, and (ii) network layer features (based on inter-arrival time, packet sizes, packet/byte counts, throughput) for QUIC flows. CSI *(Xu, 2020)* infers mobile ABR video adaptation behaviour under HTTPS and QUIC using packet size and timing information. Finally, *(Khokhar, 2019)* put YouTube under experimentation and perform network traffic measurements for QoE estimation using network related features, as well. *(Ghiëtte, 2019)* demonstrates that it is possible to utilize cipher suites and SSH version strings to generate unique fingerprints for brute-forcing tools used by an attacker.

Network middleboxes or client-side software that aim to inspect encrypted traffic can operate by acting as proxies. The common procedure is to terminate and decrypt the client- initiated TLS session, analyze the HTTP plaintext content, and then initiate a new TLS connection to the destination. *(Goh, 2010)*, Experimenting with an intrusion detection system for encrypted networks. *(Goh V. T., 2010)* propose mirroring the traffic to a central intrusion detection system, which will be able to decrypt the traffic and perform deep packet inspection, yet, without any privacy preserving guarantees. As Symantec states “most cyber threats hide in SSL/TLS encryption” (which takes up to 70% of all network traffic)[25]. Symantec Proxies and SSL Visibility Appliance decrypt traffic to support infrastructure security and protect data privacy. More specifically, Symantec offers the Encrypted Traffic Management (ETM) tool[26] that provides visibility into encrypted traffic by decrypting part of it; however, this is a technique that could eventually cause privacy violations.

Aiming to advance the state-of-the-art, FORTH proposes an automatic signature mining method for intrusion detection in encrypted network traffic. The majority of works that inspect encrypted network traffic exploits machine learning algorithms to examine the feasibility of identifying the nature of the traffic (e.g., for network analytics or network security). FORTH’s methodology builds on these feasibility results, but focuses on establishing a procedure to effectively generate intrusion detection signatures in an automated manner.

Component Evolution

We have used publicly available datasets to produce signatures for our mechanism to detect threats in encrypted traffic. Additionally, we have further increased its capabilities by utilizing open-source tools

[25] https://docs.broadcom.com/doc/ssl-visibility-en
[26] https://www.broadcom.com/products/cyber-security/network/encrypted-traffic-management

which can verify the integrity of TLS connections. Currently it is to be integrated in two different use case scenarios.

Outlook

In the future we plan to further enhance our signature database with more diverse attacks and add more capabilities for threat detection.

4. An Architectural Framework for Trustworthiness Assurance

4.1 Trust Infrastructure Overview

To ensure trustworthiness of data flows in collaborative manufacturing environments, COLLABS will design and develop components composing the trust infrastructure, presenting an architectural framework for trustworthiness assurance. Said framework will encompass:

- The provisioning of secure elements (see subsection 4.3) that will make use of built-in secure hardware to form a root of trust and verify the integrity of the code running in the device.
- The security of data in transit, via:
  - An encrypted data flow model based on which data from IoT devices will be able to be securely transferred and processed in trusted execution devices or end cloud-services, covering all upstream and downstream data flows, and supporting an extensive toolkit of cryptographic primitives, including SMC and Homomorphic encryption.
  - A novel mechanism to analyse encrypted traffic analysis (see subsection 4.5) that will further strengthen the data flows trustworthiness by capturing attacks or anomalies inside the real network traffic of the platform and reporting them to the anomaly detection modules.
- The security of data at rest, providing privacy-preserving mechanisms for the storage and analytics, based on Comprehensive Access Control mechanisms, supported by blockchain to achieve a distributed authentication scheme (see subsection *Distributed Authentication & Authorisation*).

An overview of the above-mentioned key elements comprising the COLLABS Architectural Framework for Trustworthiness Assurance is presented in *Figure 12* below. The figure also provides references to the specific Tasks (both within WP2 and within WP3 and WP4) where the individual building blocks will be developed. It also aims to provide a visual representation of the interplay between these activities and the associated components.

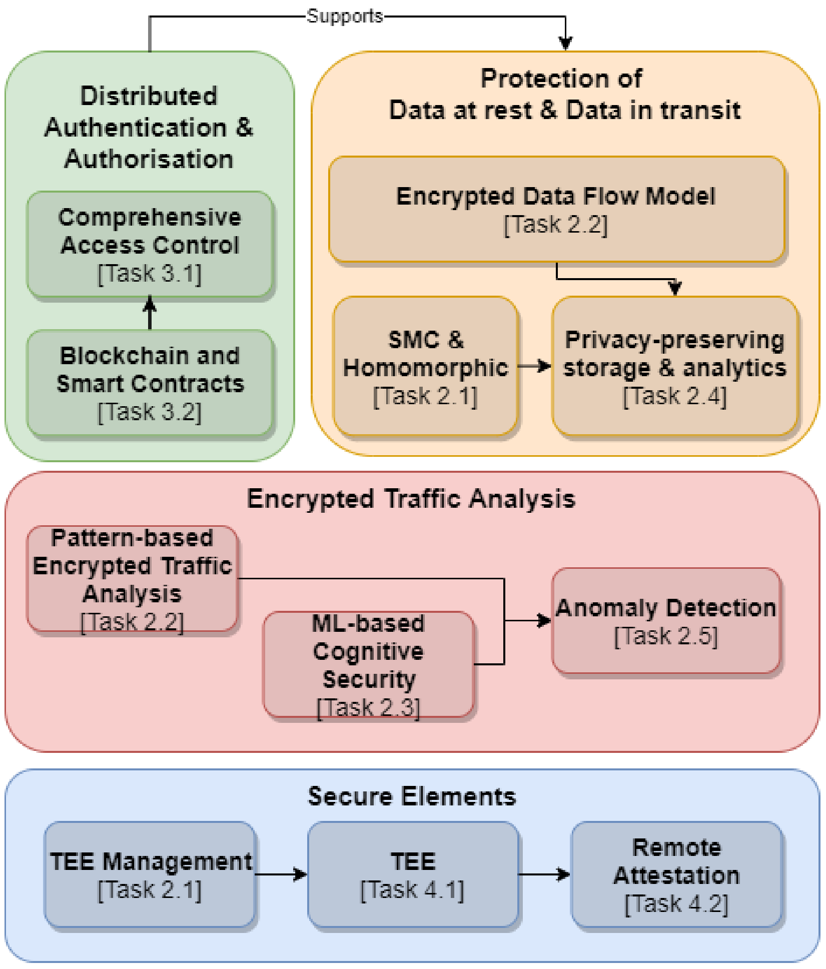

Supports
Distributed Authentication & Authorisation
Comprehensive Access Control [Task 3.1]
Blockchain and Smart Contracts [Task 3.2]
Protection of Data at rest & Data in transit
Encrypted Data Flow Model [Task 2.2]
SMC & Homomorphic [Task 2.1]
Privacy-preserving storage & analytics [Task 2.4]
Encrypted Traffic Analysis
Pattern-based Encrypted Traffic Analysis [Task 2.2]
ML-based Cognitive Security [Task 2.3]
Anomaly Detection [Task 2.5]
Secure Elements
TEE Management [Task 2.1]
TEE [Task 4.1]
Remote Attestation [Task 4.2]

Figure 12 The COLLABS Trust Infrastructure high-level architecture

4.2 Key Building Blocks

The subsections that follow provide an overview of the key features of each of the framework's key building blocks, including – where needed - pointers to specific deliverables where more details can be found.

4.3 Secure Elements

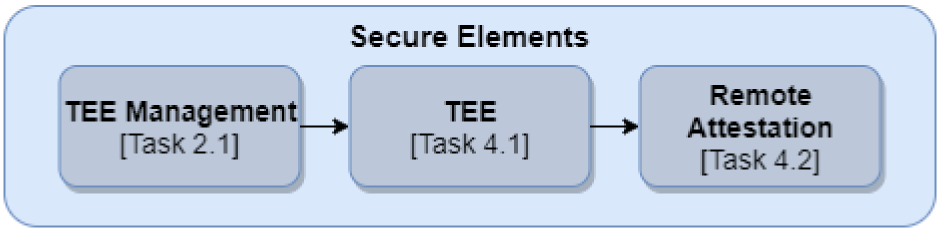


Figure 13 Secure Elements' enablers.

At the lower level of the trust architecture, COLLABS integrates trust enablers at the hardware level, leveraging the presence of trusted components on the platforms to be deployed in the manufacturing environment. These are detailed in the subsections that follow.

Trusted Execution Environments

In order to account for the increased requirements for securing a number or primitives and computations in an expanding interconnected “world” of systems, modern processors of all levels and complexities have started to integrate Trusted Execution Environments (TEEs). In server and cloud environments where high-performance devices, commonly Intel x86-64 architecture-based CPUs are used, the most prevalent TEE technology is Intel SGX. In the embedded computing space, dominated by low-power devices, ARM architecture is the most widely used and the ARM TrustZone extensions are used in order to build a Trusted Execution Environment.

The Intel SGX technology can be used to protect selected code and data from disclosure or modification. It is an ideal solution for untrusted environments, such as commodity cloud environments in which the user has minor control over aspects like the storage of the data. SGX can guarantee privacy even with an untrusted service provider.

At its root, Intel SGX is a set of CPU instructions that can be used by applications to set aside private regions of code and data. They allow user-level as well as operating system code to define private regions of memory, called enclaves, whose contents are protected and unable to be either read or saved by any process outside the enclave itself, including processes running at higher privilege levels.

SGX involves encryption by the CPU of a portion of memory. The enclave is decrypted on the fly only within the CPU itself, and even then, only for code and data running from within the enclave itself. The processor thus protects the code from being "spied on" or examined by other code. The code and data in the enclave utilize a threat model in which the enclave is trusted but no process outside it can be trusted (including the operating system itself and any hypervisor), and therefore all of these are treated as potentially hostile. The enclave contents are unable to be read by any code outside the enclave, other than in its encrypted form.

In the context of COLLABS, SGX-protected docker images can be setup in the cloud. Inside the docker images, sensitive applications may be executed, and those applications can be protected by external entities which may share the server or datacentre infrastructure of the cloud provider that hosts them.

On the other side, for IoT or edge devices based on ARM processors the TrustZone technology can be leveraged in order to secure computations and the protection of sensitive data, such as encryption keys. TrustZone is built on the foundation that the security of a system is achieved by partitioning all the system’s hardware and software resources in two worlds: The Secure World for the security subsystem and the Normal World for everything else. At the hardware level, TrustZone-enabled devices include logic that ensures that no Secure world resources can be accessed by the Normal World components while architecture extensions in the ARM cores allow the concurrent execution (in a time-sliced fashion) of Normal and Secure World applications.

Typically, in the Normal world resides a Rich OS, like Android or Linux, along with different applications and services. In the Secure world, a much more restricted secure operating system that provides a Trusted Execution Environment is executed. In the context of COLLABS, one of the TEEs that is considered is Trusty, a free and open-source TEE operating system, part of the Android Open-Source Project (AOSP).

Trusty boots as part of a chain of trust or a secure boot sequence. It creates two environments in a device with different security modes: A Non-Secure Environment (NSE) for running software components in non-secure mode (i.e., the aforementioned Normal world) and a Trusted Execution Environment (TEE) that provides trusted operations and runs in secure mode enforced by hardware (Secure world). Trusty runs in the Secure world environment. The normal world OS and Trusty software operate in a client-server relationship, with Trusty as the server. All secure operations are initiated by a client application running in the non-secure environment. A trusted application (TA), in the secure world, never initiates contact with the non-secure environment.

A Trusted Application is defined as a collection of binary files (executables and resource files), a binary manifest, and a cryptographic signature. At runtime, Trusted applications run as isolated processes in unprivileged mode under the Trusty kernel. Each process runs in its own virtual memory sandbox utilizing the memory management unit capabilities of the TEE processor.

Remote Attestation

Remote attestation is one of two security mechanisms that threat protection, prevention, and monitoring functions in the COLLABS framework rely on. It is a security mechanism that provides evidence of device reliability - a protocol that allows proving software integrity to a remote verifier, which provides evidence that the software has not been modified. This is achieved by signing protected memory areas.

For instance, in a situation when potential attackers have physical access to an IoT device, they may attempt to tamper with device firmware and try to attack it and plant some malicious code that could pose a threat for privacy and security of the entire COLLABS framework. Remote attestation offers a solution to prevent such attacks by remotely checking the integrity of the IoT device's firmware, comparing it to the known original and offering guarantees for its integrity.

Challenges that need to be addressed before remote attestation can be included in the COLLABS framework include:

- the question of current remote authentication schemes not considering the interaction between devices, and
- the question of scalability in case of usage of many devices and applying remote authentication schemes without compromising system performance.

4.4Data Protection

Data Protection enablers within the COLLABS trust infrastructure, as developed within WP2, will consider both data in transit and data at rest. These will be supported by the distributed authentication & authorisation mechanisms developed in the context of WP3. These enablers and their interplay are depicted in *Figure 14*.

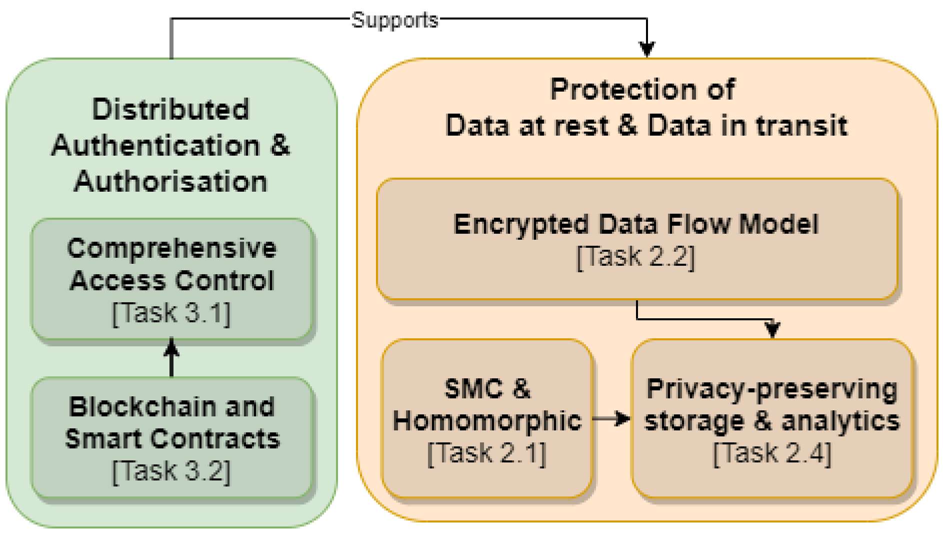

Supports
Distributed Authentication & Authorisation
Comprehensive Access Control [Task 3.1]
Blockchain and Smart Contracts [Task 3.2]
Protection of Data at rest & Data in transit
Encrypted Data Flow Model [Task 2.2]
SMC & Homomorphic [Task 2.1]
Privacy-preserving storage & analytics [Task 2.4]

Figure 14 Data protection enablers.

Encrypted Data Flow Model

An integral part of the COLLABS trust infrastructure is the definition of an encrypted data flow model that specifies the use of cryptographic primitives available within COLLABS, based on which data from IoT devices will be able to be securely transferred and processed in trusted execution devices or end cloud-services.

The model encompasses standard encryption mechanisms and best practices (e.g., for end-to-end encryption over public networks) but also innovative primitives developed within COLLABS (e.g., SMC and Homomorphic encryption).

An overarching aim of the specification of the data flow model is to consider and protect all upstream and downstream data flows (device to gateway, gateway to cloud, C&C channel). In this context, and in order to consider all such flows and the intricacies of each Use Case environment, a detailed analysis of the UCs, their scenarios, and the associated requirements (as defined within D1.2 and D1.3) was carried out, followed by consultation with UC owners (namely ALES, PCL, and REN). Through this process, the key flows needed to support each UC and its associated scenarios were identified and are depicted in *Figure 15*. Said figure and the derivatives that will be presented in the subsections that follow, also include the Purdue model layers, for reference and to maintain homogeneity with the descriptions provided in D1.2 and D1.3. If a scenario is omitted, it means that no data flows are specified for that scenario at this stage.

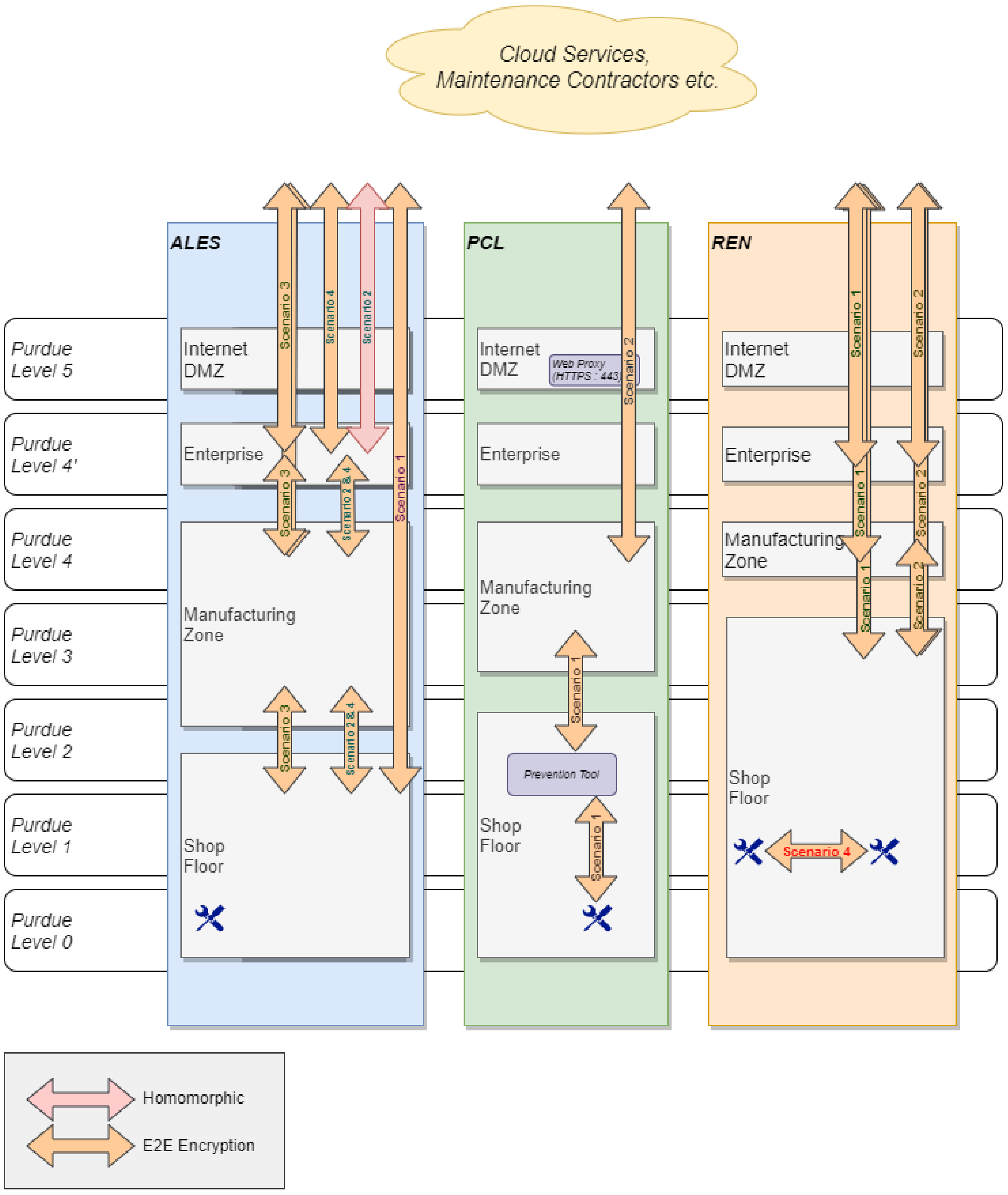


Figure 15 Overview of key data flows in Use Case scenarios

The data flows are shown in terms of the Purdue Enterprise Reference Architecture levels, as well as the corresponding layers of each of the UC environments, classified per scenario (see label within arrows), and characterised in terms of the need to support standard or homomorphic cryptographic primitives.

Following this high-level specification of the data flows, a more detailed specification must be provided for each of the UC environments and the corresponding scenarios. To this end, an analysis of available options (e.g., Information Flow diagrams, Data Flow diagrams) in the context of the COLLABS needs was carried out and the use of Information Flow diagrams was selected as the most appropriate means of defining said model, as it provides the needed level of granularity and the means to specify cryptographic primitives and their settings/configuration to protect data flows, without requiring a high level of specificity for the applications that may be supported from the specific data flows.

Nevertheless, it should be noted that the Data Flow Model will be a "live" component, constantly being updated and refined as: (a) the COLLABS solution matures from a technical perspective, and (b) the UC environments and associated scenarios are specified in more detail, paving the way for the onsite COLLABS deployment and real-life demonstration (see WP6). Therefore, as the project matures and the model evolves, the appropriateness and limitations of the Information Flow-based specification will be constantly assessed and, if the need arises, the Data Flow Model may be transferred to a different format, e.g., in a more application-specific or even machine-processable format, (through the definition of the associated grammar).

The subsections that follow provide more details on the Data Flow Model specification for the different UCs and associated scenarios.

ALES Industrial Environment

Four scenarios are identified within the ALES industrial environment that feature data flows with strong security requirements. These are highlighted in the subsections that follow.

ALES Scenario 1 - Controlled and secure remote maintenance

This scenario focuses on remote equipment maintenance from external/third party services, with support from local staff. The external/third-party services access the network and equipment, with temporary authorisations and strong segregation with respect to existing functions and data. More details on the scenario are provided in D1.2 (Section 2.b.ii.1). The data flow model for this scenario is shown in **Erreur ! Source du renvoi introuvable.** where is depicted the point-to-point flow connecting the external/remote maintainer from outside of the corporate network (cloud app) to the shop floor target machinery. This entire flow will be encrypted from end-to-end exploiting state-of-the-art protocols supporting standard encryption schemes such as HTTPS and TLS1.3. Last but not least, in case the aforementioned encryption schemes do not suffice, then additional access capabilities could be provided by making use of symmetric algorithms based on the SSH protocol and a set of predefined cryptographic policies.

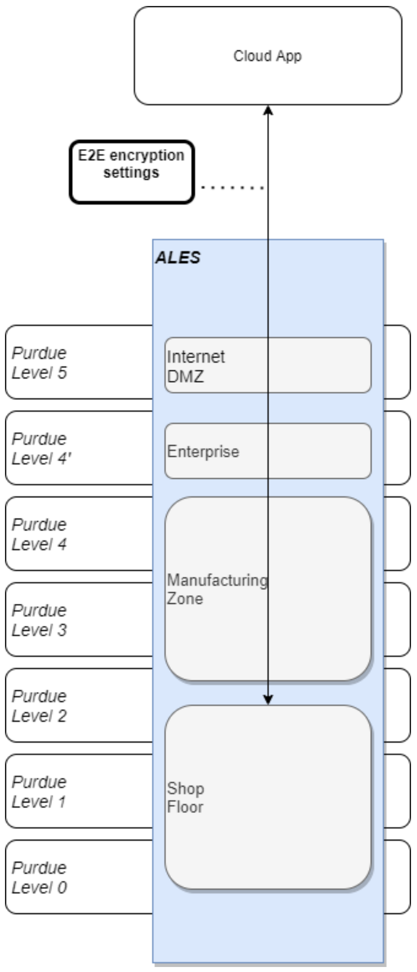


Figure 16 ALES Scenario 1 (Controlled and secure remote maintenance) data flow model.

ALES Scenario 2 - Controlled share of compliance data

Compliance data is collected from the manufacturing process and is used to evaluate and testify on the quality of manufactured goods. This compliance check may be carried out by cloud-based quality check services. In such instances, design data shall be granted to not be leaked to other ongoing computations and, whenever possible, it would be preferable to guarantee data encryption while at rest and under analysis to avoid loss of intellectual property or technical data. More details on the scenario are provided in D1.2 (Section 2.b.ii.2). The data flow model for this scenario is shown in *Figure 17* and it will leverage a homomorphic encryption methodology during compliance checks to enable the secure access, retrieval and exchange of sensitive information. Moreover, if the data is shared outside of the company to exploit third party/cloud services homomorphic encryption is applied to guarantee confidentiality of the data not only while at rest but also while under analysis. Taking into consideration the aforementioned solution, data at rest and data under analysis remain always encrypted and are kept confidential at all times, while data in transit are further wrapped within a TLS-based communication protocol as an additional layer of protection. At this point, it is worth noticing that the different zones met in this scenario may include several legacy devices, which will require the support of different but still secure interconnections and protocols.

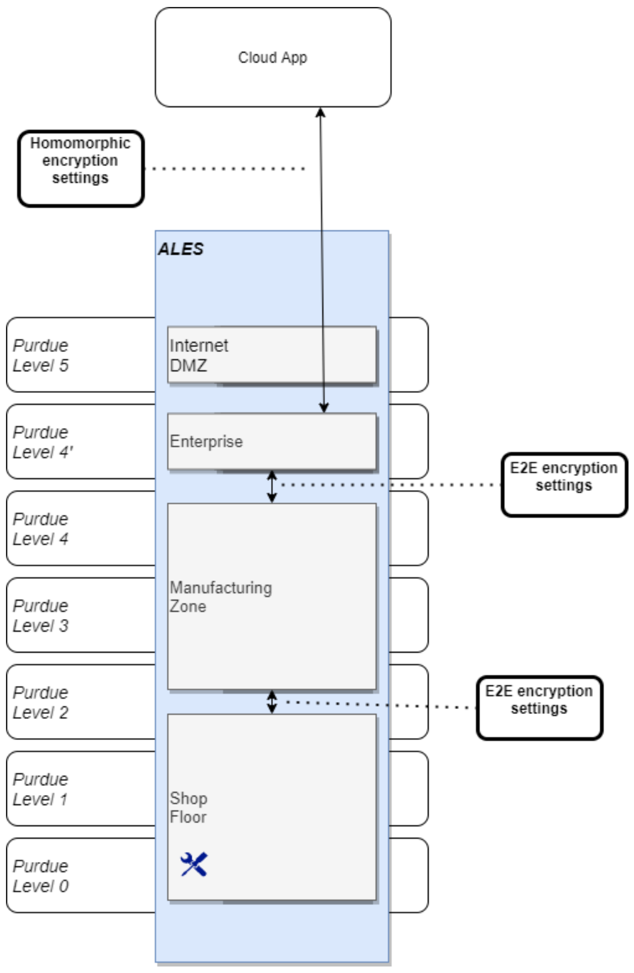


Figure 17 ALES Scenario 2 (Controlled share of compliance data) data flow model.

ALES Scenario 3 - Trusted compliance data share across the supply chain

The manufacturing of complex and safety-critical systems requires collaboration between supply chain parties. Automation of these complex information flows requires to establish trust, guarantee integrity, support traceability and accountability, potentially without relying on a single central authority. Typically, a customer requests a batch of parts with quality/compliance information attached, and the enterprise receives this request, dispatching the production order to the manufacturing zone. During production, the shop floor sends process data to the data integrator, while the set of information about the product and its supplies shared with the customer through the internet. More details on the scenario are provided in D1.2 (Section 2.b.ii.3). The data flow model for this scenario is shown in *Figure 18* where we can notice that each communication from zone to zone is securely encrypted. This scenario will be implemented using a blockchain solution based on the open-source framework Hyperledger Fabric, adopting thus all the native capabilities of blockchain technology including but not limited to protection from information tampering, traceability, and trustworthiness of dataflow. Moreover, once the information flows outside of the company a strong encryption scheme must be applied to ensure data confidentiality. For that reason, the communication between all components of this network shall take advantage of the latest TLS security protocol, namely the TLS 1.3.

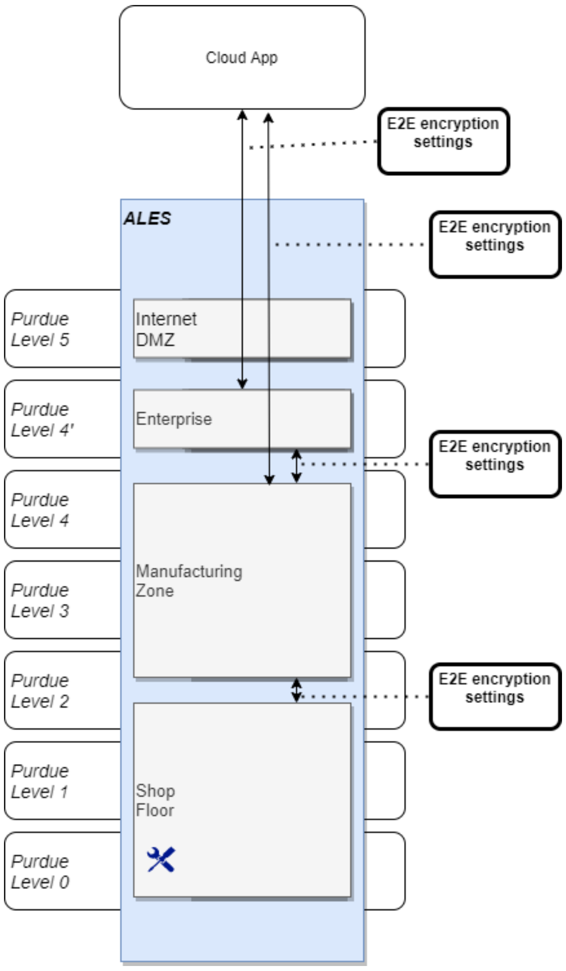


Figure 18 ALES Scenario 3 (Trusted compliance data share across the supply chain) data flow model.

PCL Industrial Environment

Two scenarios are identified within the PCL industrial environment that rely on specific data flows with intrinsic security requirements. These are highlighted in the subsections that follow.

PCL Scenario 1 - Shop floor threat detection and prevention

This scenario focuses on the shop floor and its connection with the factory infrastructure. Networking in a shop floor environment is not managed by PCL due to organizational and political reasons. This lack of insight introduces a significant level of risk and therefore a security appliance is introduced between the shop floor and manufacturing zone. In COLLABS this solution is expected to perform detection and prevention of malicious activity, and all communications to/from the factory floor will go through this tool, while it will be responsible for encryption, authenticity and integrity of the data received from the shop floor and send it to the manufacturing network. More details regarding this scenario are provided in D1.2 (Section 2.a.ii.1), illustrating a typical shop floor environment which is usually consisted of an unmanaged network of several machines and controllers. All data flow in this level of the network are unencrypted and often consist of highly vulnerable legacy protocols and environments. Therefore, in order to efficiently deal with the trustworthiness of data flows as shown in *Figure 19*, it is deemed necessary the deployment of a firewall solution in high-risk situations, as well as the execution of advanced threat detector mechanisms.

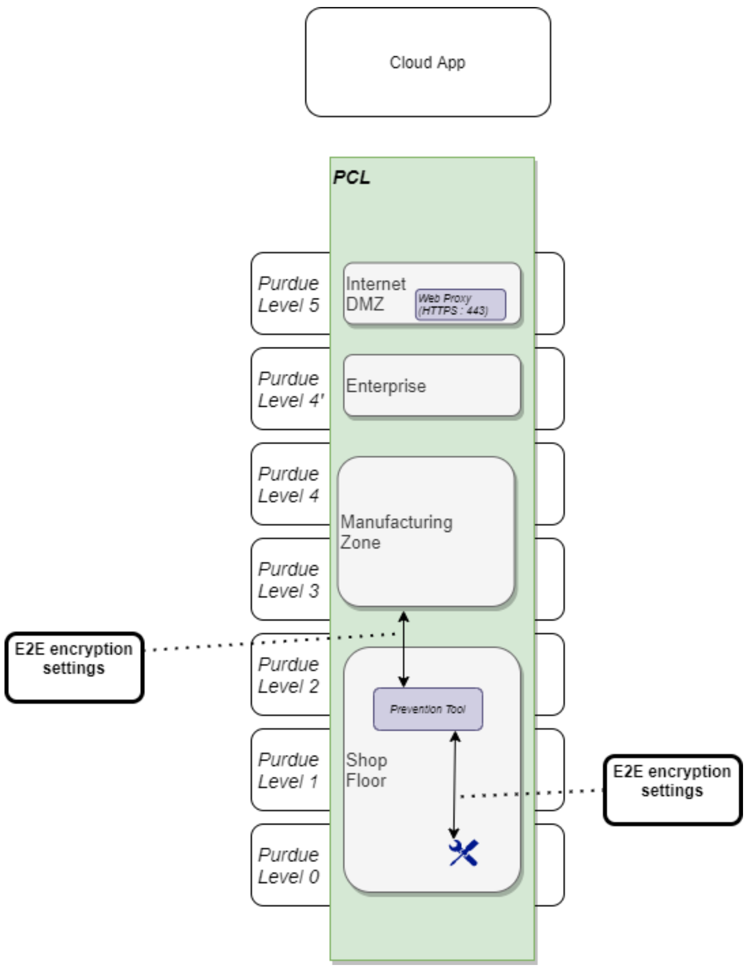


Figure 19 PCL Scenario 1 (Shop floor threat detection and prevention) data flow model.

PCL Scenario 2 - Remote data sharing

This scenario revolves on remote data sharing (i.e., sharing with external/third parties) for collaboration purposes. Interactions with equipment and tool suppliers to develop new manufacturing processes, and collaboration with consortium partners in European, national as well as regional innovation projects, require a secure data exchange solution, for confidential and business critical data. Said solution must be flexible, work bidirectionally, and be able to deal with all sorts of (near real-time) machine and process data. More details on this scenario are provided in D1.2 (Section 2.a.ii.2) while its expected data flow model is illustrated in *Figure 20*. The transmission of data in a remote data sharing environment shall take advantage of the HTTPS protocol as a medium to simulate both a batch-based data transfer and a streaming data transfer. Last but not least, another security constraint that should be taken into consideration lies to the deployment of an explicit web proxy for proxying the aforementioned HTTPS traffic.

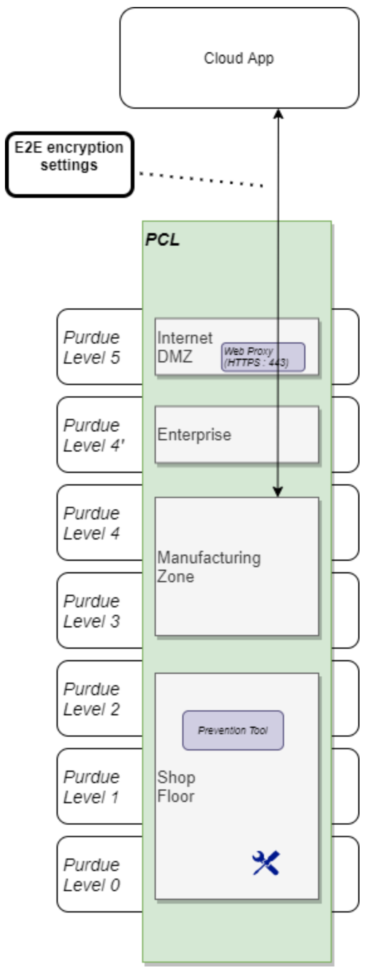


Figure 20 PCL Scenario 2 (Remote data sharing) data flow model.

REN Industrial Environment

Three scenarios are identified within the REN industrial environment that rely on specific data flows with intrinsic security requirements. These are highlighted in the subsections that follow.

REN Scenario 1 - Controlled and secured remote maintenance

This scenario involves the execution of maintenance operations, setting up production chains, and supervision of industrial processes in a remotely manner. For this, remote access to all levels of the Purdue model is required. Nevertheless, this remote access must guarantee a strong segregation with respect to existing functions and data, and especially ensure the integrity and availability of the critical production process. It should also guarantee the confidentiality of the data in the production environment, as the maintenance operations must be realized without revealing any sensitive data. More details on the scenario are provided in D1.2 (Section 2.c.ii.1). The data flow model for this scenario is shown in *Figure 21*, where the information exchanged between each zone and Purdue level share the same types of privacy-preserving and encryption methodologies. A VPN connection with REN infrastructure shall be deployed upon the Purdue Level 5, making use of the IKEv2/Ipsec NAT-T protocol and an AES-256_SHA1 cipher for connection outside of REN's intranet. On the other hand, internal connections will be encrypted with the help of the HTTPS (TLS v1.2), SSH v2, and RDP protocols.

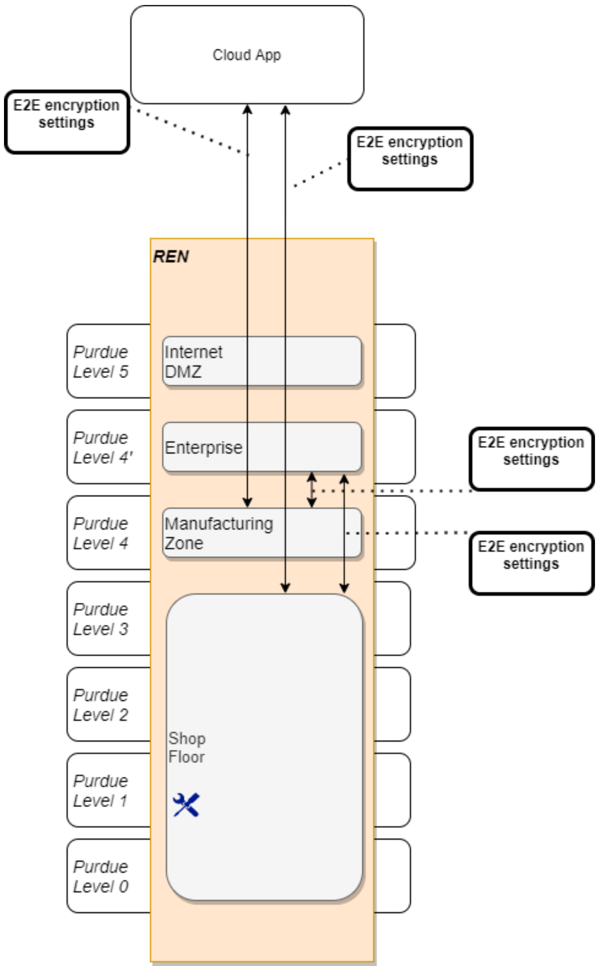


Figure 21 REN Scenario 1 (Controlled and secured remote maintenance) data flow model.

REN Scenario 2 - Cloud-based architecture for industrial processes

This scenario is motivated by the cloud-based smart factory of Industry 4.0 paradigm, modifying the production process to adopt a cloud-based agent approach to create an intelligent, collaborative, and more flexible industrial environment. Various partial use cases can be derived in this regard, as resources and services are progressively hosted externally in private/public cloud providers. An integral challenge is the need to collect more and more data from the manufacturing process, which are generated and collected for different domains of the industrial process regarding maintenance, performance, production, quality, and compliance. They key areas of impact are the shop floor level, as well as the manufacturing and enterprise zones. More details on the scenario are provided in D1.2 (Section 2.c.ii.2). The data flow model for this scenario is shown in *Figure 22* where any information being exchanged between cloud app and manufacturing zones will be going through web and reverse proxies, allowing thus the adoption of proxy-compatible encryption standards like the HTTPS TLS protocol. Dataflow being output by shop flour will be additionally filtered and whitelisted, while dataflows coming from the manufacturing zone shall support the messaging and security traits defined by the Industry 4.0 vendor-independent communication protocol known as OPC UA.

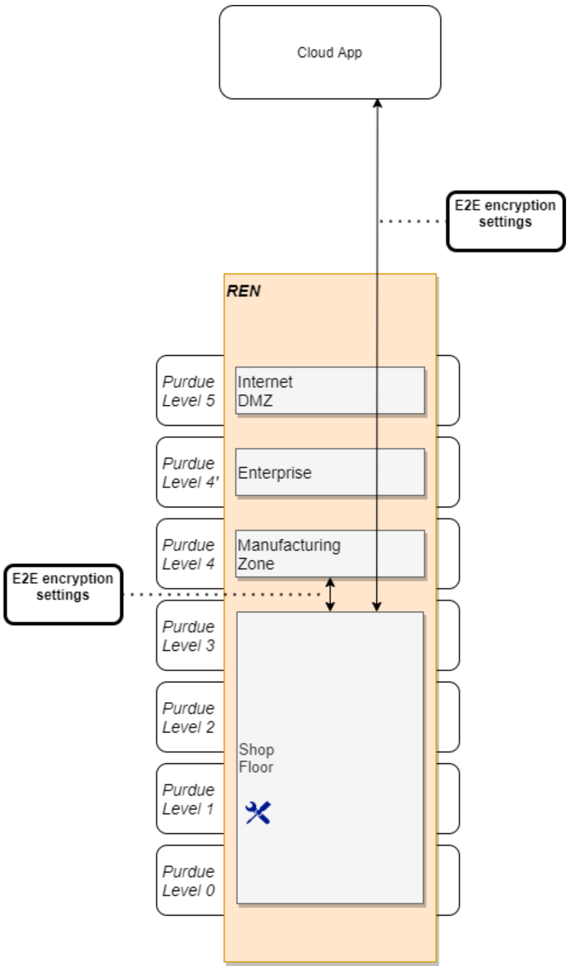


Figure 22 REN Scenario 2 (Cloud-based architecture for industrial processes) data flow model.

REN Scenario 4 - Security of connected devices

The objective of this scenario is to increase the security of the industrial environment providing security to low-cost IoT (sensors, actuators, etc.) and industrial automation devices. To achieve this, it is essential to identify the hardware security components, the cryptographic mechanisms, and the protocols pertinent to each device, while respecting their resource constraints and intricacies. Of importance is the usability of the chosen mechanisms, the provision of an effective convergence between OT and IT, with adapted protocols, and the adoption of distributed, collaborative protocols with mutual authentication mechanisms between devices to allow for automated segmentation based on devices security policies, and the security management of the devices throughout their lifecycle. More details on the scenario are provided in D1.2 (Section 2.c.ii.4). The data flow model for this scenario is shown in *Figure 23* and involves the collaboration of devices belonging on the shop floor for distributed threat detection, mutual authentication, or remote attestation purposes. This communication could adopt any encryption methodology that matches that specific IoT device, with LoRa sensors making use of LoRa encryption mechanisms, and OPC UA sensors implementing OPC UA encryption mechanisms.

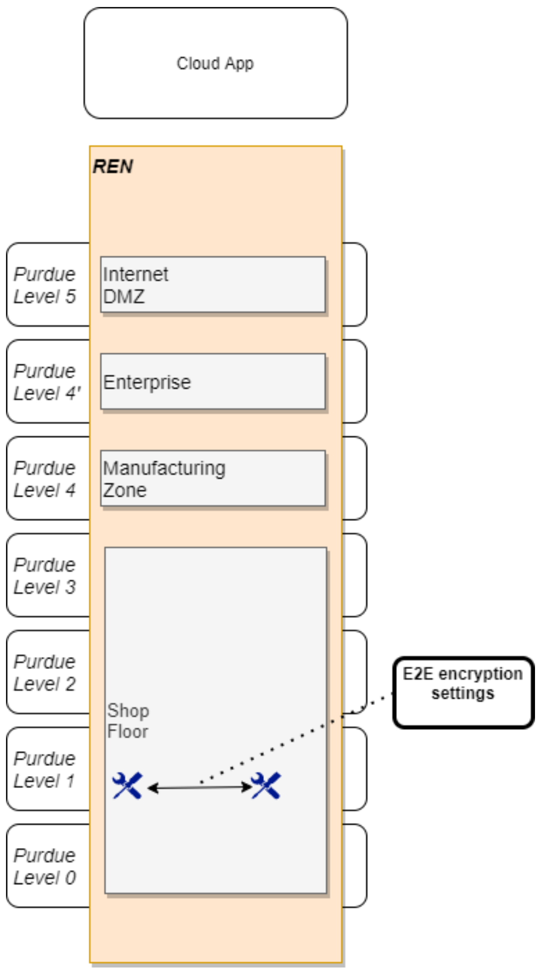


Figure 23 REN Scenario 4 (Security of connected devices) data flow model.

SMC & Homomorphic Encryption

One of the tasks of the COLLABS framework is to leverage powerful cryptographic tools, like homomorphic encryption and secure multiparty computation, to protect ML and DL algorithms and their input data's confidentiality. Data will be protected at any time as they will always remain encrypted. However, mapping complex functions onto advanced cryptographic techniques remains an ad hoc process and practically always requires cryptographic expertise to be done correctly and efficiently.

In the context of COLLABS, we provide guidelines and recommendations for hand-crafting DL solutions with advanced cryptographic techniques aiming to facilitate the work of the non-expert programmer. The AI models introduced in the COLLABS scenarios will be transformed into trustworthy AI inference systems demonstrating the effectiveness of the proposed optimizations.

Privacy-preserving Storage & Analytics

One of the tasks of the COLLABS framework is to offer privacy preserving data storage and analytics, while observing the need for easy integration and input of heterogeneous data and facilitating the adoption of collaborative analytics to enterprises. The solutions developed within the project allow - even in the MVP version of the framework - different participants to jointly process sensitive data without disclosing sensitive and confidential data to anyone outside and ensuring compliance with data access and privacy criteria.

Reliable statistical methods and machine and deep learning methods are defined and will be made available to different parties in the production chain in a privacy-preserving way. This is particularly aimed at the context of machine and deep learning where the service provider can have a model that returns predictions on the input data, and the user can get those predictions on their data without the need to expose sensitive information. This process will be further improved by applying differential privacy-based methods on shared data. With differential privacy appropriately defined, privacy guarantees are achieved by the introduction of randomness, by the data provider, in response to queries, which machine and deep learning algorithms are sent to the database. Amount of noise added to the original information is controlled by the so-called epsilon factor or privacy cost that represents a trade-off between the level of privacy achieved and the performance of a machine or deep learning algorithm. Such implementation of machine and deep learning algorithms will make it possible to configure trade-offs in terms of privacy and performance according to the needs of data owners.

Distributed Authentication & Authorisation

In a complex system such as the COLLABS framework, authentication and authorisation cannot be centralised since that would by its nature represent a security risk. Distributed authentication and authorization are achieved through a comprehensive mechanism involving various approaches including distributed PKIs, blockchain, device fingerprinting, attribute-based access control, etc. The following two subsection will overview the comprehensive access control mechanisms employed in COLLABS, as well as blockchain and smart contracts.

Comprehensive Access Control

Comprehensive access control is a data protection mechanism of the COLLABS framework. It includes security services based on trust for IoT authentication and ensures communication between trusted components only.

Communication between cloud-based applications within the COLLABS framework is managed and protected using identity and context-based access control. We use a distributed PKI mechanism to authenticate users and devices removing the need to use passwords and using certificates instead of them. Using blockchain to manage certificates will prevent tampering and ensure consistency of certificates, raising security one more level up.

On another level, the COLLABS framework contains fingerprint methods for IoT devices that may be used to detect abnormal behaviour and abnormal signals received from edge devices. We use advanced machine learning and deep learning methods to analyse the data acquired by IoT device fingerprinting to detect outliers and anomalous behaviour. Such a mechanism provides an additional level of security and confidence in industrial scenarios.

Access rules are managed centrally, at the point of policy decision-making. Using attribute-based access control (ABAC) will allow precise and flexible definition of access rules using the XACML standard. IoT devices have low processing power and low memory capacity which makes them unsuitable to act as servers. In that case, attribute-based encryption (ABE) can be used to ensure trust between the sensor and the end user. That way the IoT device only needs to encrypt the data it sends, and the policy decision point will send the decryption key to the end user. In this way, it is ensured that communication adheres to the described rules described.

Blockchain and Smart Contracts

A **distributed** ledger implements a decentralized record of transactions. A **blockchain**[27] is a particular type of distributed ledger. A **ledger** is the sequenced, tamper-resistant record of all transitions approved on the blockchain. It is a growing list of cryptographically linked records (blocks). Each block contains a **cryptographic hash** of the previous block, a timestamp, and transaction data. The blockchain is designed to resist the modification of its data. Once recorded, the data in any given block cannot be altered without needing to alter all subsequent blocks. A blockchain is usually managed by a **peer-to-peer** network where peers follow a common inter-node communication and block-validation **consensus protocol**. A **peer** is an individual participant of a blockchain network. Generally, each peer holds a copy of the ledger, which can be modified only when the majority of peers agree on it via their **consensus** protocol.

There are **public** (permissionless) blockchains, and permissioned (private) blockchains. **Permissioned** block-chains need a way to identify the participants of the network, like an identity and access management (IAM). Peers can only read and propagate transactions they are entitled to. There are different distributed ledger **platforms**, like Ethereum, Corda, and Hyperledger Fabric[28] (HLF). In permissioned block-chains like **HLF** the consensus protocol can be changed and adapted to fit specific use cases via policies. For instance, you can restrict which nodes need to verify the transactions and which nodes do not, and you can use consensus protocols that do not require costly mining.

We value distributed ledger technologies for their cryptographically granted security properties, like **immutability** and **accountability** (non-repudiation). Parties, even if they do **not trust each other,** can collaborate **without** the need for **trusted third parties**. A blockchain allows untrusted parties to collaborate under a strict and formal agreement for assurance. Distributed ledger technologies often (like when no party controls more than 50% of the nodes) can ensure the following properties:

Immutability means that once a transaction is stored on the ledger it cannot be deleted or modified any longer. **Transparency** is given as peers must accept all interactions with the ledger, they can observe and confirm each transaction and only validated transactions are written to the blockchain. An immutable and transparent ledger allows for **auditability**. For **accountability** (non-repudiation), an IAM system identifies participants in the ledger network and allows tracking their actions. In case one needs to resolve a dispute, the accountable party can be traced.

Each peer node contains a copy of the distributed ledger. Peer-to-peer data sharing guarantees **redundancy** of the database and allows checks for **consistency**. Due to this redundancy, failure of one node does not affect the **availability** of the remaining system. **High availability** may be achieved. A blockchain represents a distributed system with the potential for supporting high Byzantine fault tolerance. Due to peer consensus of the transactions, the system can be regarded as **reliable** and **fail-safe**, even if some peers default.

Blockchains offer a decentralized, immutable, and verifiable ledger that can record transactions of digital assets. Different blockchain flavours offer different scalability, security, and potential **privacy** issues. Examples of privacy issues are transaction **linkability**, **data privacy**, and GDPR compliance. Research takes place in the area of resolving privacy issues by allowing blockchain users to act anonymously, to take control of their personal data, and to keep private data confidential *(J. Bernal Bernabe, 2019).*

[27] https://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html
[28] https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html

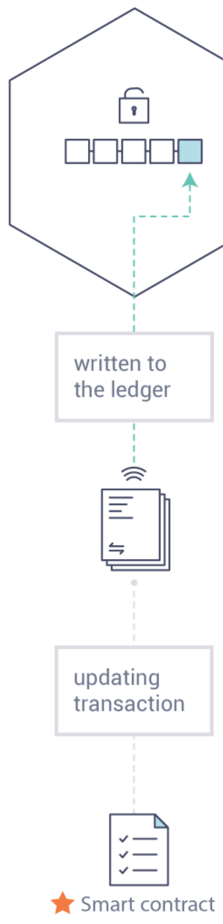


Figure 24 Smart contracts being written to the ledger.

A **smart contract** is a self-executable program running on the blockchain. It can be seen as a **stored procedure**. A smart contract can update a ledger, store variables, and instantiate and invoke other smart contracts[29]. It consists of **pre-defined rules** specified between two or multiple parties. This allows the execution of distributed and automated workflows *(Stahnke, 2020)*. A smart contract is a script describing proper interaction with the ledger. A smart contract provides a well-defined set of ways by which the ledger can be queried or updated. Smart contracts allow data to be shared with integrity, accessibility, and mutability while ensuring that these data can only be manipulated according to the rules stated in that smart contract. Only the entities that agree with the smart contract conditions interact with the smart contract.

Smart contracts are often written to ensure fairness between participating entities **without** need for **a trusted third party** *(Zupan, 2020)*. The author creates the smart contract and publishes it in the blockchain. **Conditions** written in the smart contract prior to HLF could not **be changed** even by the owner after publishing it in the blockchain, but current HLF versions allow updating the smart contract. The conditions stated in the smart contract are enforced by the code and the underlying blockchain system's consensus mechanism, where the **results** are **verified** by the participating nodes. There are many languages to write smart contracts in. For instance, Ethereum supports a Turing-complete scripting language called Solidity for smart contracts. In Hyperledger Fabric, general-purpose programming languages, like Go and Node.js, can be used to write so-called chaincode.

4.5 Encrypted Traffic Analysis

Encrypted network traffic has been steadily growing in volume in recent years, compared to nonencrypted traffic. This presents a challenge to the task of acquiring insight into the nature of the traffic, since network packet payload contents are not visible.

Traditional network traffic analysis techniques use machine learning methods to identify patterns in network flows. Anomaly detection systems can identify attacks against ICS networks in a proactive manner, reporting any suspicious activities. A major disadvantage of such systems is their focus on clear-text packet payloads only. Considering the proliferation of encrypted network traffic, such approaches are bound to become obsolete. For this reason, techniques for network traffic analysis need to focus on detecting patterns in network flows based on other approaches, such as using metadata found in packet headers.

Encrypted traffic analysis in the COLLABS framework will be achieved through the interaction of tools and components developed within three project tasks (T2.2, T2.3 and T2.5), as illustrated in *Figure 25*: Pattern-based encrypted traffic analysis (Section 3.4.2), ML-based cognitive security (Section 3.1), and anomaly detection (Section 3.4.1).

[29] https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html

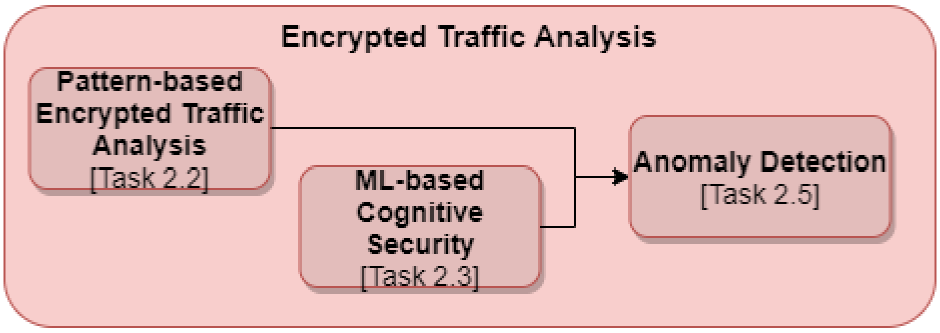

Encrypted Traffic Analysis
Pattern-based Encrypted Traffic Analysis [Task 2.2]
ML-based Cognitive Security [Task 2.3]
Anomaly Detection [Task 2.5]

Figure 25 Encrypted traffic analysis enablers.

Pattern-based Encrypted Traffic Analysis

Transport layer security (TLS) use is becoming the standard by people and organizations to protect the integrity and confidentiality of their data flows on the internet. IIoT is also in need of secure communication channels and encryption where technically possible. These encrypted channels can be misused by malicious actors to evade detection by security mechanisms such as intrusion detection systems. A novel mechanism will be used to analyse encrypted network traffic and detect attacks using signatures. It will strengthen the data flows trustworthiness by capturing attacks or anomalies inside the encrypted network traffic of the platform and reporting them to the anomaly detection modules. This mechanism will rely on metadata that can be found inside packet headers, including packet lengths, timestamps, directions, and inter-arrival times.

In the training stage publicly available datasets of malicious traffic will be collected which can be used as a base for our malicious traffic dataset. Then multiple variations of attacks will be conducted in our use cases environment and scenarios to further increase our dataset of malicious network traffic. The captured network traffic will be broken into flows and the sequences of packet payload lengths (which remain constant in multiple variations of the same attack) will be turned into signatures. These signatures will contain the fingerprint of certain actions of malicious tools and malware which target or use the TLS protocol. They rely on specific patterns or sequences of packet lengths which remain constant when these tools are used. After the generation of these signatures, they will be used to detect attacks and malicious behaviour in live network traffic and forward alerts to the corresponding modules.

In addition, CISCO’s Joy *(cisco, 2020) will be used* which can break the network traffic into flows and extract information from TLS connections such as the selected cipher suite, the server’s certificate, and the public key length. With this information extracted It will be possible to detect if a component or software is using an old and deprecated encryption method which is vulnerable to being cracked by attackers to extract sensitive information. In addition to detecting weak TLS versions, it will be able to detect TLS downgrade attacks, such as FREAK *(Beurdouche, et al., 2015)* or POODLE *(Möller, Duong, & Kotowicz, 2014)* or other vulnerabilities like Heartbleed *(Synopsys, 2020)*. By integrating CISCOS’s JOY inside the network environment it will be possible to verify the integrity and strength of every TLS connection.

In the COLLABS framework, all of the modules and devices will aim to use strong encryption but there is still a residual threat from malicious insider activity or human error. An employee could intentionally or by mistake install (vulnerable or malicious) software or bring a device which uses an older or deprecated TLS version or is infected.

ML-based Cognitive Security

The ML-based cognitive security framework (described in more detail in Section 3.1), consisting of two interacting components – the IoT Secure Wireless fingerprinting component and the ML S(N)C optimization and GMS tools – in its initial utilization, will provide increased security at the IoT device level, by enabling automated device identification based on various behavioural features and patterns, as well as threat identification by detecting anomalous device behaviour. The machine learning tools and models developed within the ML-based cognitive security framework can readily be applied to other types of network data to learn to distinguish regular from anomalous patterns in encrypted network traffic.

Anomaly Detection

Anomaly detection as has been done in the past is not ideal for the IIoT model. Installing an IDS system on a device or a group of devices provides each IDS system with a local security view. A new approach will be a distributed and lightweight anomaly detection which will leverage innovative ML techniques covering a wider spectrum and be able to detect attacks on both single devices and network wide.

In addition, several supervised and unsupervised anomaly detection methods will be considered, including methods based on clustering (K-means, DBSCAN), Isolation Forest, Local Outlier Factor, Double Median Absolute Deviation, and Mahalanobis based approaches. This approach will be better suited for an IIoT environment and will be able to detect specific security threats and identify malicious behaviours. More details about the anomaly detection module can be found in subsection 3.4.1.

5. Summary and Conclusion

In this deliverable, a detailed description of the results of the work within work package 2, until M18, is given. The results have been structured and presented as part of 1st release of the COLLABS platform.

As described earlier in the document, the COLLABS framework is organized into three levels of security:

- Level 1 - Hardware-enabled and device-level security,
- Level 2 - Inter-device level security based on distributed ledger technologies, and
- Level 3 – Machine learning-based cognitive security level.

Various technologies presented and demonstrated within the deliverable comprise the initial version of the COLLABS Level-3 security package for secure digital supply networks.

Work package 2 includes the following six tasks that round up development of the Level-3 security components in accordance with the project requirements.

- T2.1 Tools and methods for secure data sharing,
- T2.2 Trustworthiness of data flows,
- T2.3 Machine learning-based cognitive security framework,
- T2.4 Statistical Analytics and Machine- / Deep-Learning on shared data,
- T2.5 Distributed anomaly detection for Industrial IoT and
- T2.6 Workflow-driven security for supply chain and compliance in manufacturing.

Level-3 security components developed within these tasks include tools and services for:

- secure data sharing,
- trustworthiness of data flows in collaborative manufacturing environments,
- machine learning-based cognitive security framework applied to shared data,
- workflow-driven security framework for supply chain and manufacturing based on distributed ledger.

Security requirements for the COLLABS framework defined within Level-3 include monitoring different data flows originating from connected objects - IIoT devices. For this, we are using advanced cognitive security and anomaly detection mechanisms such as model-based verification and machine and deep learning to detect anomalies. Moreover, COLLABS focuses on edge-based AI solutions and analysis of encrypted network flows.

Encrypted traffic analysis mechanisms based on metadata extraction (that in turn also leverage machine and deep learning techniques applied to datasets obtained by IIoT device fingerprinting) is another requirement that has been addressed by the components of the machine learning based cognitive security framework.

A component offering secure computation outsourcing, using advanced cryptographic techniques, like (fully) homomorphic encryption, has been implemented. HE techniques can protect shared data both at rest and in use.

Outlook

Until M26, we will have a Level 3 development completed: ongoing development/finalization of components; (2) testing – i.e., validating the CSRs defined; and (3) further integration between the components described in this deliverable.

In more details: (1) The Machine Learning-Based Cognitive Security Framework will be used to address the task of Confidential Data Discovery, and it will be further investigated in collaboration with the use case providers. (2) 3ACEs aims to advance to TRL7 at the final release. (4) The HE component will be improved in terms of efficiency, by using Single Input Multiple Data (SIMD)-like implementations and by leveraging TEE components (like the INTEL SGX). (5) The WSDF tool will be further integrated into the ALES lab environment. (6) For the network monitoring, the signature database used for encrypted traffic analysis, will be enhanced, while the Distributed Anomaly Detection will be improved in terms of accuracy and privacy preservation.

The final version of the components will be presented in M26 and discussed within the deliverable D2.3 meant to describe the final phase of development within Work Package 2.

List of Abbreviations

Abbreviation	Translation
CSR_XX	Common Security Requirement no. XX (as described in deliverable D1.2)
IIoT	Industrial Internet of Things
JSON	JavaScript Object Notation
WDSF	Workflow-driven security framework
(F)HE	(Fully) Homomorphic Encryption
LHE	Levelled Homomorphic Encryption
MPC	Multi-Party Computation
SIMD	Single Instruction Multiple Data
HE	Homomorphic Encryption
SMC	Secure Multiparty Computation
MS SEAL	Microsoft Simple Encrypted Arithmetic Library
BFV (RNS)	Brakerski/Fan-Vercauteren (Residue Number System)
CKKS (RNS)	Cheon-Kim-Kim-Song (Residue Number System)
HELib	Homomorphic Encryption Library
BGV	Brakerski-Gentry-Vaikuntanathan
HEAAN	Homomorphic Encryption for Arithmetic of Approximate Numbers
NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers
ML	Machine Learning
DL	Deep Learning
ML S(N)C	Machine Learning Structured (Non)Convex Optimization
DAD	Distributed Anomaly Detection

Bibliography

Amoli, a. V. (2016). *Unsupervised network intrusion detection systems for zero-day fast-spreading attacks and botnets*. nternational Journal of Digital Content Technology and its Applications, *(pp. 1-13).*

Anderson, B. a. (2017). *Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity*. 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining *(pp. 1723-1732). ACM.*

Beurdouche, B., Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Kohlweiss, M., Pironti, A., . . . Zinzindohoue, J. K. (2015). *A Messy State of the Union: Taming the Composite State Machines of TLS*. 2015 IEEE Symposium on Security and Privacy. *San Jose: IEEE.*

cisco. (2020). joy. *Retrieved from A package for capturing and analyzing network flow data and intraflow data, for network research, forensics, and security monitoring.: https://developer.cisco.com/codeexchange/github/repo/cisco/joy/*

Conti, M. e. (2016). *Analyzing android encrypted network traffic to identify user actions*. IEEE Transactions on Information Forensics and Security, *114–125.*

Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., . . . Mytkowicz, T. (2019). *CHET: an optimizing compiler for fully-homomorphic neural-network inferencing*. Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2019) *(pp. 142-156). New York: ACM.*

Fernández-Caramés, T. M., & Fraga-Lamas, P. (2019). *A Review on the Application of Blockchain to the Next Generation of Cybersecure Industry 4.0 Smart Factories*. IEEE Access, *pp. 45201-45218.*

Ferrante, O., Benvenuti, L., Mangeruca, L., Sofronis, C., & Ferrari, A. (2012). *Parallel NuSMV: A NuSMV Extension for the Verification of Complex Embedded Systems. In* Computer Safety, Reliability, and Security *(pp. 409-416). Springer.*

Ferrante, O., Marazza, M., & Ferrari, A. (2016). *Formal Specs Verifier ATG: a Tool for Model-based Generation of High Coverage Test Suites*. Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016).

Fournier-Viger, P., Wu, C.-W., Gomariz, A., & Tseng, V. S. (2014). *VMSP: Efficient Vertical Mining of Maximal Sequential Patterns*. Advances in Artificial Intelligence. Canadian AI, 8436, *pp. 83-94.*

Ghiëtte, V. H. (2019). *Fingerprinting tooling used for {SSH} compromisation attempts*. International Symposium on Research in Attacks, Intrusions and Defenses (RAID), *(pp. 61-71).*

Goh, V. T. (2010). *Experimenting with an intrusion detection system for encrypted networks.* International Journal of Business Intelligence and Data Mining, *172–191.*

Goh, V. T. (2010). *Intrusion detection system for encrypted networks using secret-sharing schemes.* International Journal of Cryptology Research.

J. Bernal Bernabe, J. L.-R. (2019). *Privacy-Preserving Solutions for Blockchain: Review and Challenges.* IEEE Access, 7, *164908-164940. doi:10.1109/ACCESS.2019.2950872*

Kasinathan, P., & Cuellar, J. (2019). Securing Emergent IoT Applications. In Engineering Trustworthy Software Systems: 4th International School, SETSS 2018, Chongqing, China, April 7--12, 2018, Tutorial Lectures *(pp. 99--147). Springer International Publishing.*

Kelley, M., Gaehtgens, F., & Data, A. (2020). Critical Capabilities for Privileged Access Management. *Gartner. Retrieved from https://www.gartner.com/en/documents/3988410/critical-capabilities-for-privileged-access-management*

Kern, A., & Anderl, R. (2019, June 10). Securing Industrial Remote Maintenance Sessions using Software-Defined Networking. 6th International Conference on Software Defined Systems*, pp. 72-79.*

Khokhar, M. J. (2019). From Network Traffic Measurements to QoE for Internet Video. IFIP Networking Conference *(pp. 1-9). IEEE.*

Lee, J., Azamfar, M., & Singh, J. (2019). A blockchain enabled Cyber-Physical System architecture for Industry 4.0 manufacturing systems. Manufacturing Letters, 20*, pp. 34-39. doi:https://doi.org/10.1016/j.mfglet.2019.05.003*

Lotfollahi, M. e. (2019). Deep packet: A novel approach for encrypted traffic classification using deep learning. Soft Computing.

Mangeruca, L., Ferrante, O., & Ferrari, A. (2013). Formalization and completeness of evolving requirements using Contracts. Proceedings of the 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, (pp. 120-129).*

Marazza, M., Ferrante, O., & Ferrari, A. (2014). Automatic Generation of Failure Scenarios for SoC.

Mazha, M. H. (2018). Real-time video quality of experience monitoring for https and quic. IEEE INFOCOM 2018-IEEE Conference on Computer Communications *(pp. 1331–1339). IEEE.*

Mirsky, Y. e. (2018). Kitsune: an ensemble of autoencoders for online network intrusion detection.

Möller, B., Duong, T., & Kotowicz, K. (2014). This POODLE Bites: Exploiting The SSL 3.0 Fallback. *Google. Retrieved from https://www.openssl.org/~bodo/ssl-poodle.pdf*

Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., & Sadeghi, A. (2019). DÏoT: A Federated Self-learning Anomaly Detection System for IoT. 39th International Conference on Distributed Computing Systems (ICDCS)*, 756-767. doi:10.1109/ICDCS.2019.00080*

Nicolás Rosner, I. B. (2019). Profit: Detecting and Quantifying Side Channels in Networked Applications. Network and Distributed Systems Security (NDSS) Symposium*. San Diego.*

Niyaz, Q. W. (2016). arXiv:1611.07400*. Retrieved from arXiv preprint.*

Orsolic, I. e. (2016). Youtube QoE estimation based on the analysis of encrypted net- work traffic using machine learning. IEEE Globecom Workshops (GC Wkshps) *(pp. 1-6). IEEE.*

Papadogiannaki, E. e. (2018). OTTer: A Scalable High-Resolution Encrypted Traffic Identification Engine. International Symposium on Research in Attacks, Intrusions, and Defenses. (RAID) *(pp. 315-334). Springer.*

Parmisano, A., Garcia, S., & Erquiaga, M. J. (n.d.). Stratosphere Laboratory. A labeled dataset with malicious and benign IoT network traffic.

R. Cammarota, e. a. (2020). Trustworthy AI Inference Systems: An Industry Research View. ArXiv, cs.CR, 2008.04449.

Sherry, J. e. (2015). Blindbox: Deep packet inspection overencrypted traffic. ACM SIGCOMM Computer communication review *(pp. 213–226). ACM.*

Shone, N. e. (2018). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence*, 41-50.*

Stahnke, S. (2020). Workflow Enforcement for Certification of the Construction of Industrial Plants. *Passau: University of Passau.*

Synopsys. (2020). heartbleed.com. *Retrieved from The Heartbleed Bug: https://heartbleed.com/*

Tang, T. A. (2016). Deep learning approach for network intrusion detection in software defined networking. International Conference on Wireless Networks and Mobile Communications (WINCOM) *(pp. 258-263). IEEE.*

Taylor, V. F. (2017). Robust smartphone app identification via encrypted network traffic analysis. IEEE Transactions on Information Forensics and Security*, 63-78.*

Xu, S. S. (2020). CSI: inferring mobile ABR video adaptation behavior under HTTPS and QUIC. Fifteenth European Conference on Computer Systems*, (pp. 1-16).*

Yuksel, S. E., Wilson, J. N., & Gader, P. D. (2012). Twenty Years of Mixture of Experts. IEEE Transactions on Neural Networks and Learning Systems, 23*(8), 1177-1193.*

Zupan, N. K. (2020). Secure Smart Contract Generation based on Petri Nets. *Munich, Passau. doi:10.1007/978-981-15-1137-0_4. 2020*

6. Appendix. Methods for Screening and Improving the Security of Future COLLABS Solutions

6.1 Methods and Guidelines for Oblivious Machine Learning

6.1.1 Advanced Cryptography Tools

In Task 2.1, we leverage powerful cryptographic tools (homomorphic encryption, secure multiparty computation) to protect the ML and DL algorithms. As described in D1.4, in COLLABS a DL outsourcing scenario will be adopted, as a locally running DL model will be migrated to the Cloud. The main security goal is to protect the confidentiality of both the input data and/or the model.

ALES is working on the refinement of P&W Scenario #2 (from Deliverable D1.2) to define the requirements for the application of HE. In particular, ALES is considering the case where a compliance dataset and analysis algorithm are deployed on the cloud for product compliance analysis (images, geometrical measures, sensor data).

Given our use case scenario, we have decided to protect the model execution with Fully Homomorphic Encryption (FHE) as the FHE schemes have limited communication overhead compared to SMC ones which are very interactive. The main Dl operations that will be implemented are:

1. Convolution
2. Matrix multiplication
3. Activation functions: ReLu.
4. (Max) Pooling

Since 2009, the performance of FHE computations has impressively improved with a speedup of 10^9. However, programming FHE applications remains a demanding task that requires cryptographic expertise *(R. Cammarota, 2020)*. The non-expert programmer can use the multiple available implementations (there are several open-source libraries), but as has been reported, the produced code can be 10^3 times slower than the code written by an FHE expert.

In Task 2.1, we provide recommendations and guidelines on the optimal usage of the available schemes. Emphasis will be given to the ML and DL models used in COLLABS. The main goal is to optimize the trade-offs between security, performance and accuracy and facilitate the programmer in managing the encryption parameters, the noise level, and the data approximation.

6.1.2 Writing a FHE Program

The initial program (plaintext program) can be re-written as a FHE program by naively replacing all the computations with the corresponding FHE ones (mainly additions and multiplications). However, this approach does not scale and can be extremely slow (more than 10^3 times).

The general structure of a FHE circuit appears in the following Figure 7.

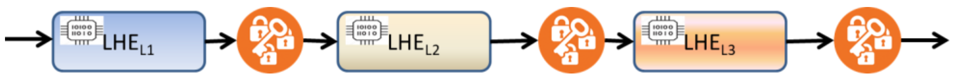


Figure 26 The FHE circuit structure.

All the proposed FHE schemes can compute any program. However, in order to achieve that, they need a rekeying operation called bootstrapping (orange circles in the scheme). This operation is very costly and the FHE programmers must avoid it if possible. The boxes between the bootstrapping operations are called Levelled HE (LHE) circuits. This part implements a circuit with a given multiplicative depth L.

That is that - the programmer with a LHE can implement any plaintext program with multiplicative depth up to L. However, the larger the value of L, the slower the program becomes. Thus, the programmer must find the optimal combination of the values L1, L2, etc., such that the overall complexity is the minimum possible. Such optimization is still an open problem and strongly relies on experience.

In what follows, we provide guidelines and recommendations for the unexperienced programmer to create an efficient FHE program. Emphasis will be given to the implementation of efficient LHE programs. The programmer needs to make several decisions, including the selection of scheme, the specific data layout (for parallelism) and the encryption parameters, among others. The LHE program must be secure, efficient, and correct.

6.1.3 Selection of the FHE Scheme

While the first FHE schemes only supported Boolean operations (resulting to rather huge circuits), the last generation of FHE designs are able to homomorphically perform integer operations. This allows us to evaluate arbitrary arithmetic circuits efficiently without having to "explode" them into Boolean circuits. Among them, the most efficient ones are BFV, BGV and CKKS. All of them are constructed on the Ring Learning with Errors (RLWE) problem.

Practical applications, like the DL model inference, require floating point arithmetic and approximate computation. However, all the FHE schemes support only integer operations. Thus, in order to obliviously compute using these schemes, we use fixed-point arithmetic and we explicitly define a scaling factor. While we can use all three FHE schemes with fixed-point arithmetic representation, only CKKS (and its variant RNS-CKKS) can give practical programs. The problem is the fast increase of the scaling factor with multiplication. In order to cope with that, we need to select huge encryption parameters, and especially an extremely large plaintext-coefficient.

However, CKKS offers another alternative, as it enables ciphertext rescaling. It allows the programmer to reduce the scaling factor, at the expense of less accuracy. Actually, this is exactly the paradigm of floating-point arithmetic. We are going to use CKKS in COLLABS.

All three FHE schemes support four main evaluation operations:

1. addition of ciphertexts,
2. addition of ciphertext and plaintext,
3. multiplication of ciphertexts and
4. multiplication of ciphertext and plaintext.

The FHE schemes implement nicely polynomial computations. For applications that use non-polynomial operations (like exp, log, etc.), it is assumed that such functions can be approximated by polynomials. It has been demonstrated that this is possible for ML and DL.

Another important parameter is batching compatibility. In order to compute in parallel operations on messages, data can be processed in SIMD (single instruction multiple data) like way that allows a vector of input messages to be encoded in a single plaintext. With batching, the operations take place in an element-wise fashion increasing efficiency. All three FHE are batching compatible.

6.1.4 FHE Libraries

There are several open-source libraries that implement the three main schemes. In Table 1, we provide the main ones. We want to emphasize that the field is very dynamic. Theoretical advances and new improved implementations change the scenery rapidly.

Library	***HE schemes***
MS SEAL	*BFV (RNS), CKKS (RNS)*
HELib	*BGV, CKKS*
Palisade	*BGV, BFV, CKKS*
HEAAN	*CKKS*

Table 3 FHE open-source libraries.

CKKS is the most suitable scheme for ML and DL applications and there are two versions of the scheme. The first is referred to as plain "CKKS," while the second is known as "RNS-CKKS." Both have open-source implementations, like the HEAAN, PALISADE[30] and SEAL libraries. We will mainly use the last one, as it offers more efficient implementations that can use machine-sized integer operations as opposed to multi-precision libraries. However, it imposes further restrictions on the circuit multiplicative depth. Thus, we will have to decide on a per use case.

6.1.5 FHE Security

Each FHE is initialized with a set of parameters. The value of these parameters determines the level of security, the efficiency, and the accuracy of the scheme. In COLLABS, we will first decide on the security level, and then we choose the other parameters. The level of security will be at least 128 bits (security parameter).

For the same level of security, the encryption parameters affect the performance and the accuracy of the generated FHE program. Typically, larger values lead to more accuracy, but slower programs, while smaller values to faster and less accurate programs. Choosing an adequate set of parameters is tricky.

Current FHE schemes work by introducing noise during encryption that is subsequently removed during decryption. The amount of noise introduced during encryption and every intermediate operation is controlled by a set of encryption parameters that are set manually. Setting these parameters low can make the encryption insecure. On the other hand, setting them large can increase the size of encrypted text and increase the cost of homomorphic operations. Moreover, when the accumulated noise exceeds a bound determined by these parameters, the encrypted result becomes corrupt and unrecoverable.

The parameters must be chosen following the recommendations of the "HomomorphicEncryption.org," the open consortium of industry, government, and academia for the standardization of homomorphic encryption. Following the white paper[31] (section 5.4) that they maintain, the parameters must be chosen following the tables provided.

6.1.6 FHE Correctness

[30] *https://palisade-crypto.org*
[31] *http://homomorphicencryption.org/white_papers/security_homomorphic_encryption_white_paper.pdf*

In all FHE schemes, noise is added in every homomorphic operation, and this noise must be controlled in order to perform correct decoding at the end. The CKKS scheme introduces an additional challenge by providing approximate results (of course with much higher efficiency). There are two extra sources of error: the error from the initial encoding of the input values and the noise from the rescale operation.

FHE Error

All the evaluation operations increase the level of noise. The level of noise must be controlled and always remain under a certain bound for correct computation. There are two main approaches. Either we increase the level of tolerance, i.e., the acceptable upper bound of noise, by increasing the ciphertext coefficient, or we reduce the level of the produced noise periodically during the computations. More precisely:

- The upper bound of acceptable noise depends on parameter Q, the ciphertext modulus. This parameter can be increased to afford noise increase, but at the expense of reduced efficiency.
- The programmer can apply a modulus **switching operation**. This operation reduces the level of noise by modifying the ciphertext modulus. The modulus is divided by one of its factors. However, the modulus switching operation can only be applied a limited (predefined) number of times and it is a costly operation.
- Limit the number of multiplications. The ciphertext multiplication operation increases the level of the noise exponentially. Thus, it is necessary to choose computations that minimize the multiplication depth of the circuit to be evaluated.
- Use bootstrapping to reset noise. This operation re-encrypts the ciphertext homomorphically and it is a very expensive operation.

Selecting the Scaling Factor

Neural network models typically use floating-point values that must be transformed to fixed point and determining the optimal fixed-point scaling factors to use is not straightforward. The use of fixed-point arithmetic with scaling factors introduces two problems:

1. The encoding can be lossy.
2. The scaling factor increases very fast with multiplications,

Determining the scaling factors to use for the inputs and the output is difficult as it involves a trade-off between performance and output precision. This is **an open problem**.

Using high enough scaling factors allows errors to be hidden. However, the computations become shortly expensive as the scaling factor increases. Furthermore, power functions computation would rapidly overflow modest values of the modulus Q, requiring impractically large encryption parameters to be selected. In order to control the size of the scaling factor, CKKS supports an operation called rescaling.

Rescaling operation

CKKS schemes support an operation called rescaling that scales down the fixed-point representation of a ciphertext. Practically, it truncates the fixed-point representation. Consider a ciphertext x that contains the encoding of 0.25 multiplied by the scale 2^{20}, the rescaling operation will truncate to encode the value at a scale of 2^{10}.

Rescaling has a secondary effect of also dividing the ciphertext's modulus Q by the same divisor as the ciphertext itself. That means that there is a limited "budget" for rescaling built into the initial value of Q. This is similar to the modulus switch operation. However, the modulus switch does not affect the scale factor, only it brings down the level of a ciphertext.

Regarding the RNS-variants of CKKS that is supported by SEAL, the truncation is performed by dividing the encrypted values by prime factors of Q (it is the product of r distinct prime integers). The order of the prime factors is fixed, i.e., the order of the divisions is fixed. That means that from any point there is only one valid divisor available for rescaling. The value of Q is reduced by this factor each time.

The rescale must also be lower bounded with respect to the output. If the output scale is less than a value, the computed output probably loses accuracy, and this is irreversible. This complicates selecting points to rescale, as doing so too early might make the fixed-point representation so small that the approximation errors destroy the message. The user decides on the input and output scale. Also, the maximum allowed rescale factor can be defined. In some implementations this maximum value is dictated by the library, like in the SEAL library (2^{60}).

Input encryption coefficients correctness

To apply any homomorphic operation, the two input ciphertexts must be at the same level, i.e., must have the same modulus Q. Furthermore, all additions (and subtractions) require that the input ciphertexts are encoded at the same fixed-point scale. A homomorphic evaluation is correct, when:

- **Rule 1 (addition):** Two ciphertexts that are added must have the same modulus coefficient and the same fixed-point scale factor.
- **Rule 2 (multiplication):** Two ciphertexts that are multiplied must have the same modulus coefficient (not the same fixed-point scale factor, necessarily).

CKKS also supports a modulus switching operation. The application of the modulus switching operation modifies the current modulus Q and brings down the level of a ciphertext without scaling the message. Inserting the appropriate rescaling and modulus switching operations to match levels and scales is **a non-trivial task**.

Relinearisation

A ciphertext is initially represented by two polynomials. However, the multiplication of two ciphertexts (consisting of several polynomials each) yields the ciphertext of the result with more computations than the initial ones (if q and w are the input, then q+w+1 polynomial will be the result). To set a bound on the number of polynomials an expensive operation, called relinearisation, is performed. Relinearisation reduces the number of polynomials to the input value (i.e., q+w) and using a distinct public key.

Selecting the optimal placement of relinearisation operations is proven to be **an NP-hard problem**. Usually, relinearisation takes place after every multiplication so that a ciphertext needs always only two polynomials to be represented. However, it is an open problem to find other (better) strategies, given that the decryption operation can be generalized easily to the m-polynomials representation, with m>2.

6.1.7 FHE Efficient Implementation

The cost of individual homomorphic operations are orders of magnitude more expensive than the equivalent plaintext operations. Thus, we must compute them in parallel as much as possible. At the same time there is another weapon in our arsenal, the so-called batching.

Batching

Amortising the cost of FHE operations requires utilizing the vectorization capabilities (also called as “batching” in the FHE literature) of the FHE schemes. Data can be processed in a SIMD-like way by encoding a vector of input messages in a single plaintext. With batching, the FHE operations take place in an element-wise fashion increasing efficiency. The SIMD width in CKKS and RNS-CKKS is N /2 and they support rotation operations that mimic the shuffle instructions of the SIMD units. The rotation by i requires a different public key for each i. We usually produce keys for all the powers of 2 (until N). Also, they do not support random access to get a particular slot value from the ciphertext. We need a multiplication with a plaintext for such operations.

However, vectorization is **not a straightforward procedure**. Different mapping on to ciphertext vectors results in different circuits with different performances. More precisely, for each resulting circuit the programmer must determine the encryption parameters that maximize performance while ensuring security and correctness. We will propose cost-based guidelines to assist the programmer in systematically searching over the different options.

6.1.8 Guidelines and Recommendations

We provide general guidelines for the programmers based on the analysis above. It is still necessary to have a good understanding of HE and its limitations. First, we provide guidelines for an implementation that does not use 'batching.' We recommend an unexperienced programmer to get familiar with such designs. Then, we provide guidelines regarding message vectorization.

'Simple' Approach for LHE Circuit Design

- ✓ Choose the appropriate FHE scheme. We assume that CKKS will be the selection for DL inference.
- ✓ Select the version of plaintext circuit with the minimum multiplicative depth
- ✓ Select the input and output level of accuracy.
- ✓ Select the maximum rescale factor (in SEAL this must be 2^{60}).
- ✓ Replace all the operations (additions and multiplications) with the corresponding FHE ones.
- ✓ Use relinearisation after each FHE multiplication.
- ✓ Use rescale, if needed, after each multiplication.
- ✓ Verify that the input data of all operations has the correct input encryption coefficients. Place the modulus switching operation where necessary.
- ✓ Compute the value of the modulus coefficient Q. Q is either power of 2 or the product of r prime numbers. We want to minimize either log Q or number r.
- ✓ Use the table of coefficients to get the minimum possible value N.
- ✓ Estimate the performance of the circuit using asymptotic complexity (Table 2). Be careful that we can get only a rough estimation as the hidden coefficients have a massive impact. Always it is better to measure the complexity of the actual implementation.

HE Operation	*CKKS*	*RNS-CKKS*
addition, subtraction	$O(N \cdot \log Q)$	$O(N \cdot r)$
scalar multiplication	$O(N \cdot \log Q)$	$O(N \cdot r)$
plaintext multiplication	$O(N \cdot \log N \cdot M(Q))$	$O(N \cdot r)$
ciphertext multiplication	$O(N \cdot \log N \cdot M(Q))$	$O(N \cdot \log N \cdot r^2)$
ciphertext rotation	$O(N \cdot \log N \cdot M(Q))$	$O(N \cdot \log N \cdot r^2)$

Table 4 Asymptotic complexity of the main CKKS HE operations. M(Q) is the complexity of multiplying large integers32.

- ✓ Implement all the operations at the same level in parallel.
- ✓ Free memory when possible.

Vectored Approach

LHE efficiency can be significantly improved using batching. The 'Simple' LHE must be rewritten and all data must be organized in vectors. Such an implementation is suitable for neural network inference, where all operations are tensor operations. The programmer must first design the vectored version of the tensor circuit and then apply the guidelines described above.

Note: Batching can be applied in simple LHE as well without any modification. It covers the case that we have several input data for prediction.

For the vectored circuit:

- ✓ The rotation operation needs a different public key for each l-position cyclic shift of the message.
- ✓ The SIMD width in all CKKS versions is N/2. That is that - we can store up to N/2 messages in a plaintext.
- ✓ The mask operation increases multiplicative depth.
- ✓ There is a trade-off between splitting a large tensor and increasing N to fit it in.
- ✓ There are several choices of layouts. I.e., several different ways to store the messages in a plaintext slot (rewrite input as a vector).
- ✓ Depending on the data layout, different operations must be applied. The best layout selection depends on the complexity of each operation per FHE scheme.

6.2 Distributed Anomaly Detection for Industrial IoT

6.2.1 Overview / Functional Description

Industrial Control Systems (ICS) are systems that are part of Operational Technologies (OT). Their functions are management, monitoring and control of infrastructure and industrial processes. This applies to manufacturing plants, industrial plants, electrical power and energy systems and other critical infrastructures. In the past, these systems were not intended to be accessible remotely. They remained in closed and controlled environments and were isolated from corporate networks and external networks. Thus, protocols used by such systems implemented little or no security features. Lately, in order to improve their efficiency, the ICSs integrated communications technologies from the world of Information Technology (IT). They are also more and more open and connected to services on corporate networks or on external networks. This openness brings great flexibility and benefit in the management of industrial processes. It also enables the development of new high added-value functions for industrial infrastructures. On the other hand, it exposes these infrastructures to new attack surfaces, such as cyber-

[32] *CHET: an optimizing compiler for fully-homomorphic neural-network inferencing. R Dathathri et al. Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2019*

attacks that were specific to IT technologies. Especially since legacy ICS protocols were not developed to deal with these attacks.

Intrusion Detection Systems (IDS) are applications that monitor a network or system in order to detect attacks and malicious behaviour that takes place there. In particular, Network Intrusion Detection Systems (NIDS) are IDSs used to monitor networks.

A NIDS collects network traces and analyses them to detect attacks or intrusions into the monitored network. The NIDS performs Deep Packet Inspection (DPI), and extracts information about the upper layers of the OSI models in the packet capture. This analysis can have low latency constraints. In these cases, the performance of NIDS is particularly important as it must analyse large amounts of information and return information in a very short time. In some cases, these time constraints can be relaxed, and more extensive processing can be performed.

Different intrusion detection techniques exist. The signature-based methods of attacks use a knowledge base of existing and already listed attacks. These methods recognize the characteristics of the known attacks in the analysed network packets. Signature-based methods require access to a set of attack signatures. With such methods, it is not possible to detect an attack that is not listed in the set of signatures.

There are also anomaly-based intrusion detection methods. These methods aim to detect abnormal behaviour or communications between network components. They often use machine learning algorithms to determine the normal behaviour of the network. It is then possible to detect deviations from this normal behaviour in the network. Anomaly-based IDSs may detect zero-day and unknown attacks.

ICSs have specificities such that the usual IT security measures cannot be applied systematically:

- There are often constraints about response time between devices, which make it difficult to use cryptography mechanisms to ensure data integrity and/or confidentiality.
- These time constraints may also disallow using simple filtering equipment, as it may introduce too much latency.
- Protocols used in ICSs such as Modbus or DNP3 are not secured by design, and a corrupted node inside the network may not be detected. It may produce illegitimate frames in order to disrupt the functioning of the ICS.
- As ICSs operate continuously, they are not updated frequently. Often, updating such a system is done at scheduled maintenance operation, where it is shut down. Consequently, if a vulnerability on a device is referenced, it will not be corrected until the next scheduled maintenance. During this time, the ICS remains vulnerable.
- An ICS has a lifespan of several decades. Even if the future ICSs are integrating more and more cybersecurity primitives, the current and legacy systems do not necessarily have these possibilities. They are therefore exposed to vulnerabilities.

Regarding this context, IDSs are suitable for ICSs. IDSs allow enhancing the security and attacks detection on industrial systems without interfering with their operation.

There are different metrics to assess the relevance of an IDS. The most common ones are:

- the maximum throughput that the IDS can support,
- the false alarms rate,
- the detection rate,
- the coverage, which is the set of attacks that are detectable by the IDS,
- the robustness against evasion techniques and attacks directed at the IDS.

It is possible to categorize different attack types on ICS networks, depending on the scope of their targets and their applications:

- Network wide attacks are attacks targeting (or originated from) a large number of devices on the network. These attacks can evade device-wise analysis systems. For instance, a Distributed Denial of Service (DDoS) attack with low traffic generation per device may not be detected by a device-wise attack detection system. It may be necessary to look at network traffic statistics to detect these attacks.
- Single target attacks are attacks targeting a single device on the network. These attacks can fool statistical analysis systems since it can be stealthy. For instance, scanning open ports on a single target on a network with lots of devices and communication may not be detected by statistical analysis. It is necessary to look at a per-device traffic to detect these attacks.

Attacks on the network are characterized by malicious packets. These packets divert or intend to divert devices from there nominal operations. It is possible to categorize different malicious packet types:

- Data Maliciousness packets are packets that hold malicious data that intent to exploit some vulnerability at the target. For instance, command injection, malware uploading… belong to this malicious packet type. These packets may hold the malicious data at the application layer and thus may need special analysis of their transport layer's payload.
- Contextual Maliciousness packets are packets that are not malicious by themselves, but they are sent for malicious intention. For instance, flooding attacks and replay attacks use legitimate packets to disrupt the operation of the network. These packets can normally be seen in both benign and attack scenarios. The context they come in decides whether they are malicious or not.

Challenges of Distributed Anomaly Detection (Model Detection)

The application of IDS technologies to ICSs encounters several challenges. Indeed, ICSs use heterogeneous, and sometimes proprietary, communication protocols. As a result, many of these protocols are not supported by conventional IDSs. The use of encryption in application layers limits the packet analysis capabilities of IDSs.

The limited communication resources of some devices of the ICSs, their spatial distribution as well as the latency requirements of industrial processes make the collection of network traces complicated. This may result in the need to deploy multiple network probes to capture traffic. Since each probe has a capture of local exchanges, it will not have a global view of ICS communications.

ICSs systems are driven by industrial process constraints. Their communication characteristics are predictable and relatively constant. The anomaly-based methods for IDS are therefore privileged for these systems. These approaches tend to generate a high rate of false positives. Training machine learning models can also be complicated by a low volume of communication, resulting in insufficient training data.

Distributed Learning

Most distributed learning algorithms have their foundations in ensemble learning. Ensemble learning builds a set of classifiers in order to enhance the accuracy of a single classifier. Although there are other methods, the most common one builds the set of classifiers by training each one on different subsets of data. Afterwards, the classifiers are combined in a concrete way defined by the ensemble algorithm. Thus, the ensemble approach is almost directly applicable to a distributed environment since a classifier can be trained at each site, using the subset of data stored in it, and then the classifiers can be eventually aggregated using ensemble strategies. In this sense, the following are the advantages of distributed learning:

- Using different learning processes to train several classifiers from distributed data sets increases the possibility of achieving higher accuracy, especially on a large-size domain. This is because the integration of such classifiers can represent an integration of different learning biases which possibly compensate one another for their inefficient characteristics.
- Learning in a distributed manner provides a natural solution for large-scale learning where algorithm complexity and memory limitation are always the main obstacles. If several computers or a multi-core processor are available, then they can work on a different partition of data in order to independently derive a classifier. Therefore, the memory requirements as well as the execution time, assuming some minor communication overhead, become smaller since the computational cost of training several classifiers on subsets of data is lower than training one classifier on the whole data set.
- Distributed learning is inherently scalable since the growing amount of data may be offset by increasing the number of computers or processors.
- Finally, distributed learning overcomes the problems of centralized storage. Thus, many research works on distributed data learning have been concentrated on ensemble learning where the emphasis is put on making accurate predictions based on multiple models.

6.2.2 Architecture Design: High Level

In this section, we describe our distributed anomaly detection system.

Level-3 security requirements for the COLLABS framework include monitoring various data flows from IIoT devices using cognitive security mechanisms like model-based verification and machine and deep learning for detection of anomalies. Moreover, COLLABS focuses on AI-based solutions for the edge and analysis of encrypted network flows.

Trade-off between accuracy and performance will be further improved using Reservoir Computing in combination with deep learning methods with a wide range of available resources from edge to the cloud. Another requirement is related to encrypted traffic analysis mechanisms based on metadata extraction that will use machine and deep learning techniques. For wireless IoT devices, these methods will be applied to datasets acquired from wireless fingerprinting.

The system is composed of local components that are deployed on the edge of the network (Figure 8). We call these components "Local Detection Units" (LDU). Each LDU is assigned a role to analyse and detect anomalies in network traffic locally.

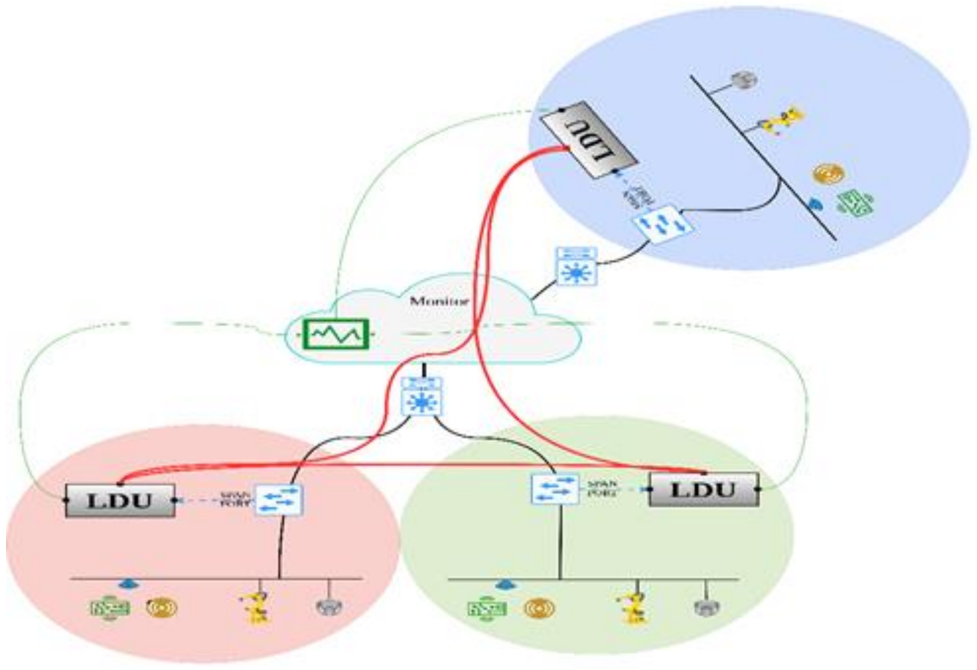


Figure 27 Distributed Anomaly Detection System.

As a black box, an LDU takes as input the raw network traces of the monitored network. Depending on the deployment scenario, the LDU can be connected to a span port that provides network traffic with (almost) no latency (Figure 8). The LDU produces decisions regarding the analysed packets. In addition, LDUs communicate with each other to share information that aims to increase the accuracy of detection. In detail, an LDU is composed of multiple machine learning (ML) models. Each of the ML models analyses different features in the analysed packet. These models start with pre-trained parameters and are updated over time using the previously monitored traffic. A **Pre-processor** component first parses the packets and extracts relevant features. The **Activator** then activates the corresponding ML model depending on the type of treated features. Each activated ML model outputs the decision and the confidence level. Finally, the **Accumulator** merges the output of the activated ML models, based on predefined rules, into a final decision. Since the models are updated, updates on the local models' parameters need to be transferred to other LDUs. Sharing these updates increases the efficiency of the learning process. We leverage distributed learning to share these model updates.

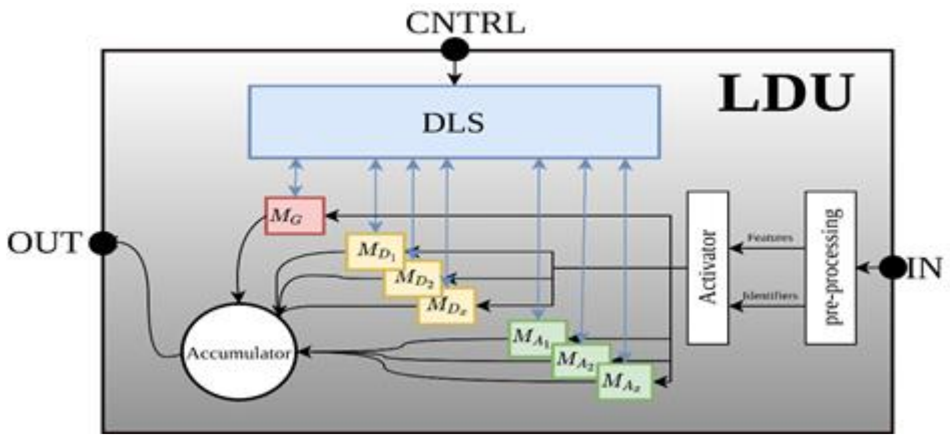


Figure 28 Local Detection Unit.

Our system manages efficiently the overhead of distributing the learning information. A Distributed Learning Service (DLS) runs on each LDU. A DLS regularly transfers and receives learning information from and to the LDU ML models.

Pre-Processing

Network traffic is mirrored toward the LDU as raw packets. The Pre-processor first extracts identification information from the packets. This information is used to choose which models are activated for this packet. The identification information may be:

- Source MAC address,
- Destination MAC address,
- TCP session,
- Application Layer Protocol.

The **Pre-processor** then parses these packets and extracts the relevant features from them. As a preliminary step to build an accurate and precise IDS, it is important to understand the potential attacks on the system. Attacks can be split into network-wide attacks and single-target attacks. Also, packets can be malicious in different ways: data maliciousness packets, and contextual maliciousness packets.

Driven by the fact that the first step to detect an anomaly is to look at the right place, and since knowing "where to look" is equally important to knowing "how to look", the step of feature extraction from raw packets is a critical step for an efficient IDS. However, not all these attacks can be revealed from the same features. Hence, we propose to categorize the types of extracted features into three categories: Packet statistical features, Packet's metadata features, and Packet's payload features. We categorize features in this fashion since each of these categories reveals information relevant about special types of attacks.

Parsing Packet Statistical Features

These features are statistical data collected over a window of N packets. They may include:

- Transmission rate,
- Reception rate,
- Packet size mean,
- Packet size standard deviation,
- Ratio of TCP packets,
- Ratio of UDP packets.

By analysing these features, we are able to detect network-wide attacks.

Parsing Packet Metadata Features

These features are extracted mostly from the headers of the Network and Transport layers. They may include:

- Source port,
- Destination port,
- Packet length,
- TCP Flags: SYN | SYNACK | ACK | PUSH | FIN,
- Protocol: TCP / UDP / Other.

By analysing these features, we are able to detect single-device attacks, especially the ones that are attained using contextual malicious packets.

Processing Packet's Payload Features

These features are extracted from the payload of the transport layer. Since this is highly dependent on the application protocol used, the pre-processor extracts the payload and handles it as raw data. Later, the respective ML model parses this raw data depending on the identified application. A mechanism for analysing encrypted network traffic might be used. It will rely on the metadata that can be found inside packet headers, including packet lengths, timestamps, direction, and inter-arrival times. By analysing these features, attacks using data malicious packets can be detected.

Activator

The Activator transfers features extracted from a packet to a specific ML model. We follow a methodology of fine-grained separation of models and creating conditions for their activation and for the features they analyse. Therefore, the Activator is designed to register handlers for each ML model deployed in the LDU. The handler contains:

- The method to call that ML model,
- The inputs of the callable ML model,
- The conditions when this model is called.

When a model is deployed, these three pieces of information need to be specified for that model. For example, model M_x can be deployed and registered such that it takes as inputs all the statistical features and is called on packets originating from devices that are of type D_x. Activator can also take into account the resource consumption of the LDU and the volume of network traces to process before deploying or not ML models.

Machine Learning (ML) Models

The ML models are the core of the system. Each ML model works independently to detect certain anomalies in an analysed packet. It runs on specific packet features passed by the Activator. It then outputs a value between 0 and 1 representing the probability of the analysed packet being anomalous. The ML models are treated as black boxes, so there are no constraints about their internal design. An example of ML models can be Recursive Neural Network (RNN) models, Deep Neural Network (DNN) models, Support Vector Machine (SVM) models, etc. Models can be added or removed from the system at any time to adapt to computing resources, to changes in the network topology, on the latency requirements and on the security requirements. To design our system, we think that three types of models are needed to detect all types of attacks mentioned in Section 4.2.1.

General Models: MG

- **Description**: Analyse the overall behaviour of the devices to detect network-wide attacks such as worms.
- **Input Features**: Packet statistical features from $\mathbb{N}$ consecutive packets.
- **Training Data**: All network packets.
- **Activation Conditions**: On receiving $\mathbb{N}$ packets.
- **Possible Designs**: Clustering (SVM, k-means), Outlier Detection (Local Outlier Factor, Isolation Forest).

Devices Specific models: MDx

- **Description**: Analyse the behaviour of a specific device on the network to detect an anomaly in the device performance with respect to the normal behaviour of devices of the same type. Thus, a model per device type is trained and a copy of the trained model is spawned for each device in the network.

- **Input Features**: Packet's metadata features.
- **Training Data**: Packets received and sent by devices of the same type of device x.
- **Activation Conditions**: On receiving packets from/to the device x.
- **Possible Designs**: RNN (Gated Recursive Units, Long Short-Term Memory), Auto-Encoders.

Application Specific Models: MAx

- **Description**: Analyse the payload data on the application layer to detect anomalies in the device's behaviour with respect to normal data used by the device's application. A model per application protocol is trained and a copy of the trained model is be spanned for each opened session.
- **Input Features**: Packet's payload features.
- **Training Data**: All the payload data for application app sent/received by all devices such app is the protocol used in session x.
- **Activation Conditions**: On receiving a packet belonging to session x.
- **Possible Designs**: NLP techniques (BERT, Transformers, RNN).

Accumulator

When a packet is mirrored to the LDU for analysis, multiple models can be activated. The Accumulator is the component that merges the output of the different activated models and gives as a result the final decision about the analysed packet. The Accumulator can be a decision tree whose parameters are trained in a supervised manner. In this case, the decision tree can be trained using real world attack scenarios.

Distributed Learning Service (DLS)

Distributed learning seems essential in order to provide solutions for learning from both “very large” data sets (largescale learning) and naturally distributed data sets. It provides a scalable learning solution since the growing volume of data may be offset by increasing the number of learning sites. Moreover, distributed learning avoids the necessity of gathering data into a single workstation for central processing.

Distributed clustering or effective voting are two strategies that can be adopted for distributed machine learning. They also can be used alone or in combination with other algorithms such as decision rules, stacked generalization, meta-learning, knowledge probing, distributed pasting votes, effective stacking, or distributed boosting,

Despite the clear advantages of distributed learning, new problems arise when dealing with distributed learning as, for example, the influence on accuracy of the heterogeneity of data among the partitions or the need to preserve privacy of data among partitions.

Addressed COLLABS Common Security Requirements

Regarding security requirements, CSR_08 is partially covered because our system provides a mean to detect anomalous behavior. Moreover, *our distributed anomaly detection system will be not part of MVP. It will be more detailed and refined in the next deliverables D2.2 and D2.3.*

6.3 Formal Specs Verifier (FSV)

Formal Specs Verifier (FSV) is an industrial-strength formal verification framework, providing an environment for analysis of complex systems designs against functional, safety and security requirements. The framework is structured into: (1) a translation layer, supporting automated and formal transformation from a system design language to a mathematical model including formal requirements specifications, (2) an algorithmic layer, where the various verification objectives are addressed by specific verification algorithms, (3) an analytic layer, where computational tasks generated by verification conditions are discharged by backend engines, such as SAT/SMT/OMT solvers, industrial and academic model checking tools, internally developed analytics for specific verification tasks. The platform has a high-maturity core, developed across the years, and is currently adopted in multiple industrial programs, and it can be extended with more explorative/research tools/analyses by leveraging its layered architecture. Extensions can include design and specification languages, new verification tasks and related algorithms, novel analysis back-ends.

Functional Description

The FSV framework has been used in several industrial and research projects, and extensions have been presented for the verification of embedded systems *(Ferrante, Benvenuti, Mangeruca, Sofronis, & Ferrari, 2012)*, synthesis of failure scenarios *(Marazza, Ferrante, & Ferrari, 2014)*, automatic test generation *(Ferrante, Marazza, & Ferrari, Formal Specs Verifier ATG: a Tool for Model-based Generation of High Coverage Test Suites, 2016)*, requirements validation *(Mangeruca, Ferrante, & Ferrari, 2013)* as well as applications to concrete industrial size cases.

In the context of COLLABS the main functionalities exposed by FSV are:

- Property verification: given a model with properties described as invariant or temporal logic formulae the tool can apply different formal techniques with the aim of verifying (proving) the property or falsifying it providing a counterexample
- Synthesis of failure scenario: given a model, the specifications of failure modes that can affect any component in the model and a set of properties that should hold regardless the system failures the algorithm provides a Minimal Critical Failures Set (MCFS) that leads to an unsafe behaviour (invalidated property).

Preconditions and Input

FSV takes as input Simulink/Stateflow models. Therefore, this is a strong precondition for its usage. Furthermore, models and the properties must be specified according to specific modelling guidelines to be fully supported by the tool. These modelling guidelines guarantee that FSV will properly preserve the semantics of the source model during translation into the target language suitable for model checking.

Postconditions and Output

The expected output of the tool is different based on the features used:

- Property verification: the assessment of a design can end with three different results. If the property is invalidated, the tool returns a counterexample that (simulated) allows to reproduce the behaviour invalidating the property. Instead, if the property gets proven the tool returns confirmation of its validity. The third possible outcome can be “unknown” which is the case if the engine cannot solve the problem (e.g., bounded model checking, or timeout reached).
- Synthesis of failure scenario: this assessment can end with three different results. A minimal set of failures that can lead the system in an unsafe state (if the property is invalidated), or a confirmation of soundness if the property is proven to be valid even under fault conditions. The third possible outcome can be “unknown” which is the case if the engine cannot solve the problem (e.g., bounded model checking, or timeout reached).

Component Design

FSV works on formal models designed in Simulink and Stateflow following proper design guidelines. The toolchain consists of four main components:

- Parser: in this phase the input is processed by using a MATLAB API-based parser that reproduces the model using an internal formal intermediate representation. During this phase, the tool detects unsupported blocks or unsafe constructs, if present.
- Model translator: the internal representation is processed and optimized with the objective of producing a formal model that encodes the input model as a finite state machine. In addition, (1) a set of assumptions on the primary input of the system (Environment model) and (2) the property to be verified are translated and processed by the optimization phase as specification patterns.
- Target language encoder: finally, this step produces artifacts that can be processed by state-of-the-art model checking tools such as NuSMV.
- Formal analyser: FSV is capable to interface different verification engines. The main used is NuSMV which is a symbolic model checker used for the verification of industrial designs.

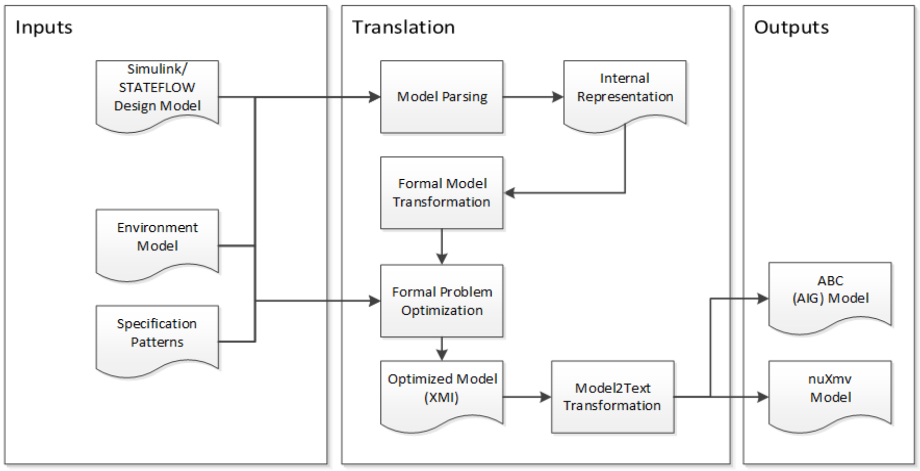


Figure 29 Formal Specs Verifier transformation flow.

Addressed COLLABS Common Security Requirements

FSV will not be deployed as part of the COLLABS framework therefore it is not directly addressing any of the Common Security Requirement (CSR) identified in D1.2.

This tool is a support utility that will be used to formally prove at design level that the solutions proposed by COLLABS are addressing properly the security requirements by verifying at design level that the required security properties satisfy the CSR.

Outlook

The development of the tool is out of COLLABS scope, in the context of this project we will just apply formal techniques exploiting FSV capabilities.