CELTIC-NEXT
Eureka Cluster

Project Report

CISSAN

Collective intelligence supported by security aware nodes

D2.3 Implementation definition of the CISSAN architecture and distributed system elements and interfaces

Editor: Niko Candelin, Netox and Ilgin Safak, University of Jyväskylä

Abstract

This report provides the details of the updated architecture of the CISSAN platform developed in the project to enhance the cybersecurity by enabling collective intelligence in the Internet of Things (IoT) and Operational Technology (OT) networks. It describes the architectural principles, the elements of the distributed system, and its interfaces. It provides the details of how the key components of the system are implemented and integrated, including the management services, the security services, the data exchange mechanisms, and the user interfaces. It provides the details of how the key components of the system are integrated and how they interact with the IoT devices and the OT systems and the external tools. It also explains the operation of the CISSAN architecture.

Project **CISSAN**

Public

April 2026

Participants in the CISSAN project are (in alphabetic order with the project coordinator first:

- University of Jyväskylä (coordinator)
- Affärsverken Karlskrona AB
- Arctos Labs Scandinavia AB
- Bittium Biosignals Ltd
- Bittium Wireless Ltd
- Blekinge Tekniska Högskolan
- Blue Science Park
- Clavister AB
- Councilbox Ltd
- Geodata ZT GmbH
- Mattersoft
- Mint Security Ltd
- Netox Ltd
- Nodeon Ltd
- Scopesensor Ltd
- Savantic AB
- Techinova AB
- Wirepas Ltd

CISSAN-Collective intelligence supported by security aware nodes

D2.3 Implementation definition of the CISSAN architecture and distributed system elements and interfaces

Editor: Niko Candelin, Netox and Ilgin Safak, University of Jyväskylä

Project coordinator: Ilgin Safak, University of Jyväskylä

CELTIC published project result

Executive Summary

The main objective of the CELTIC-NEXT CISSAN project is to improve the level of cybersecurity and cyber resilience of Internet of Things (IoT) and Operational Technology (OT) ecosystems that utilize device, edge, and cloud computing. This document describes the implementation of the revised CISSAN architecture, including the realization of the architectural decisions and the architectural design principles for the realization of the collective intelligence-based cybersecurity. It describes the implemented system components, distributed components, and interfaces, as well as the integration of management services, security services, data exchange mechanisms, and user interface components. It also describes how the new CISSAN architecture is deployed and operated for the realization of the IoT and OT ecosystems and how it interacts with devices, networks, and external tools. With the increasing trend of using IoT and OT infrastructures as entry points for carrying out large-scale cyber-attacks, the implemented architecture provides a practical and adaptive platform for the realization of collective intelligence-based cybersecurity and cyber resilience for addressing the limitations of the existing security solutions.

List of Authors (in alphabetic order according to partner name)

- Lars-Göran Magnusson, Arctos Labs
- Jari Partanen, Bittium
- Anders Liden, Clavister
- Rodrigo Martinez, Councilbox
- Klaus Chmelina, GeoData
- Ilgin Safak, University of Jyväskylä
- Pasi Tapanainen, University of Jyväskylä
- Pyry Kotilainen, University of Jyväskylä
- Stella Palenius, University of Jyväskylä
- Mikko Lehkonen, University of Jyväskylä
- Veikko Markkanen, University of Jyväskylä
- Xiaobang Sun, University of Jyväskylä
- Teemu Kemppainen, Mattersoft
- Niko Candelin, Netox
- Karoly Makonyi, Savantic
- Jyrki Portin, Scopesensor
- Oliver Bölin, Techinova

Table of Contents

Abbreviations

AAA	Authentication, Authorization, and Accounting
AI	Artificial Intelligence
API	Application Programming Interface
CI	Collective Intelligence
CTI	Collective Threat Intelligence
DTW	Dynamic Time Warping
GAN	Generative Adversarial Networks
ICS	Industrial Control Layer
IIoT	Industrial Internet of Things
IoT	Internet of Things
ISP	Internet Service Provider
IT	Information Technology
JSON	JavaScript Object Notation
JSONL	JSON Lines
LoRaWAN	Long Range Wide Area Network
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
OT	Operational Technology
PLC	Programmable Logic Controller
PQC	Post-Quantum Cryptography
RADIUS	Remote Authentication Dial-In User Service
REST	Representational State Transfer
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SIEM	Security Information and Event Management
SOAR	Security Orchestration, Automation, and Response
STIX	Structured Threat Information Expression
WASM	WebAssembly Module
VM	Virtual Machine
VPN	Virtual Private Network
Wi-Fi	Wireless Fidelity protocol

Definitions

Framework: The architectural principles, concepts, methods, and interfaces that provide a structured blueprint for the organization of system components, separation of responsibility, and interaction of components in a system.

Network: The communication infrastructure, devices, systems, and services that facilitate data exchange and coordination in the IoT and OT environment. This includes field networks, gateways, switches, and transport links that interconnect devices, edge components, and CISSAN platform services.

Platform: The implementation of the framework, which includes the hardware and software components and their integration required to develop the actual system.

Solution: The integrated hardware, software, services, and processes that facilitate the implementation of the CISSAN framework. This integrates the CISSAN platform components, use case systems, and security and orchestration mechanisms to address the specific cybersecurity and resilience challenges.

1 Introduction

The CELTIC-NEXT CISSAN (Collective Intelligence Supported by Security Aware Nodes) project focuses on providing improved security and automation in the Internet of Things (IoT) and Operational Technology (OT) environments by using collective intelligence (CI), artificial intelligence (AI), and distributed systems technologies to address the increasing scale, complexity, and coordination of cyber threats against IoT and OT environments. It aims to provide improved security and automation in IoT and OT environments by transforming traditional security from a centralized and static capability to a distributed and dynamic capability that is supported across networked devices and services.

This report focuses on the implementation of the improved architecture of CISSAN, which is based on previous definitions of the architecture and architectural principles. The report provides information on how the main architectural elements are realized as distributed system components and interfaces, along with management and security services, data exchange mechanisms, and user interfaces that are used to facilitate CI-based security solutions. The report further provides information on how the improved architecture is used in practice and interacts with IoT devices, OT systems, and other tools, which provides a concrete and operational view of the CISSAN architecture and platform to project partners and other interested parties.

1.1 Objectives of the Document

The objectives of this document are to define and describe the implementation of the new CISSAN architecture, including the components of the distributed system, the interfaces, and the workflows, to provide a clear and concrete understanding of the implementation of CI-based cybersecurity for IoT and OT. This document will provide details on how the architectural components are implemented and how the architectural components are integrated and deployed at the device, edge, and cloud levels, and how they interact with the IoT devices, OT systems, user interfaces, and external tools. This document provides details on the architecture, implementation, and integration of the CISSAN platform and serves as a reference document for the project partners and stakeholders.

1.2 Scope and Structure of the Document

The objective of this document is to provide an implementation definition of the revised CISSAN architecture, with specific reference to the distributed system components, interfaces, and integration mechanisms that facilitate CI-based cybersecurity within IoT and OT environments. The document provides an overview of the architectural principles, core services, interfaces, and deployment model, along with their interaction with other entities such as devices, networks, user interfaces, and other external tools. The document also provides an overview of the working aspects of the architecture, including distributed security, orchestration, and disaster recovery. The document is arranged in such a manner that it provides an overview of the architecture and design principles, followed by core system components and interfaces, CI mechanisms, and finally implementation considerations, validation, and conclusions to assist with implementation and further development.

2 CISSAN Architecture Overview

2.1 CISSAN Platform Vision and Scope

The CISSAN platform was developed as a common integration and experimentation platform for the CISSAN project partners to implement, integrate, and validate the mechanisms developed within the project, with the various use case system representations interconnected within a shared architecture. The CISSAN platform was also developed to reflect the project's strategic objective of facilitating the gradual transition of legacy centrally managed systems towards more distributed and collaborative system architectures, as opposed to an abrupt shift towards a fully decentralized approach, which would be difficult to deploy in operational environments. The CISSAN platform was developed with a focus on the security by design approach, zero trust architecture, and aligned with the project's overall security principles to ensure the security considerations are integrated into the overall architecture of the system. The CISSAN platform provides a unified platform for the development, testing, and demonstration of CI, trust management, security orchestration, and interoperability mechanisms, enabling the evaluation of the solutions developed to be consistent across heterogeneous domains including transportation, smart grids, tunnel construction, manufacturing execution systems, and automated disaster recovery.

The CISSAN platform’s vision is to create a unified, secure, and interoperable cybersecurity architecture that enables distributed detection, collective response, automated protection, and CI across heterogeneous IoT and OT environments, while supporting a gradual transition from legacy, centrally managed systems to more distributed and collaborative architectures. It provides the architectural and implementation foundation for distributed devices, edge components, management services, and human operators to function as a CI-enabled system in which security-related events are detected locally, correlated globally, and addressed collectively in an automated and coordinated manner. It is designed with the flexibility that is needed to integrate with existing infrastructures and security systems, thereby decreasing the barriers to adoption. Moreover, there is an inherent aspect of future-proofing against new emerging security threats, including those from quantum computing, through cryptographic flexibility and the possibility of integration of post-quantum cryptography (PQC) methods. Through this approach, the operationalisation of CI becomes a core capability that transforms isolated security mechanisms into a coordinated, scalable, and resilient security system suitable for critical infrastructures and industrial environments over the long term.

The CISSAN platform provides a security model for IoT / OT environments, with four main cybersecurity functions: Detection, Response, Protection, and Intelligence, which are described below.

- Detection: The function of detecting cyberattacks in IoT and OT environments, using various methods and techniques, such as anomaly detection and signature-based (or rule-based) detection through network traffic analysis, device behaviour analysis, and user behaviour analysis. The sub-functions of detection are:
    - Meta Data: The process of extracting relevant information from network traffic, such as the source, destination, protocol, payload, and impact of data packets. Similar approaches can be applied to events in an endpoint device, such as timings, parent-child process chains, etc.
    - Labelling: The process of assigning labels to network traffic, such as normal, suspicious, malicious, or unknown, based on the analysis of relevant metadata and the comparison with baseline / normal models and profiles and threat intelligence. Similar approaches can be applied to events in an endpoint device.
    - Source / Impact: The process of identifying the source and the impact of network traffic, such as the device, the service, the vulnerability, or the threat that generated or affected the observed data packets.
    - Settings Management: The process of managing the settings and parameters of a detection function, such as thresholds, rules, policies, and alerts.
    - Reporting: The process of reporting the results and findings of a detection function, such as the metadata, labels, sources, impacts, and alerts, to the relevant stakeholders and systems, such as users, response functions, or cloud backends.
- Response: The function of responding to cyberattacks and anomalies in IoT / OT environments, using various methods and techniques, such as automated actions, manual actions, or collective actions. The sub-functions of threat responses are:

    - Self / Collective Awareness: The process of keeping aware of the current state of an IoT / OT environment, including devices, services, vulnerabilities, threats, and incidents, and sharing this information with other systems and stakeholders, such as cloud backends, the protection function, or the intelligence function.
    - Automated Response: The process of executing appropriate predefined actions to mitigate or prevent cyberattacks, including blacklisting devices, isolating devices, executing operating system processes, performing disaster recovery, applying patches or updates to devices or services.
    - Reducing Attack Surface: The process of reducing the exposure and the risk of an IoT / OT environment, such as disabling or removing unnecessary or unused devices, services, or protocols, or enforcing secure configurations and policies for devices and services.
    - Deny / Restrict: The process of denying or restricting access or the communication of devices or services, including performing authentication, authorization, encryption, as well as applying access rules, firewall rules, and whitelisting and blacklisting policies.
    - Configuration: The process of configuring and tuning the settings and parameters for the response function, such as actions, rules, policies, and alerts for the sub-functions of automated response, attack surface reduction, or deny / restrict.
- Protection: The function of protecting an IoT / OT environment from cyberattacks, using various methods and techniques, such as device security, network security, or cloud security. The sub-functions of protection are:
    - Identify posture improvement: The process of identifying and assessing the current security posture of an IoT / OT environment, including devices, services, vulnerabilities, threats, and incidents, and suggesting improvements and recommendations to enhance the security level and performance. Improvements may include isolating devices with low-trust scores from the network, updating their security state in the device ledger, triggering remediation processes via the orchestrator such as marking the device as blacklisted, thereby preventing it from performing security tasks and requesting a new task distribution for the network, performing disaster recovery, and enforcing access control policies at the network level.
    - Initiate change: The process of initiating and implementing changes and improvements to an IoT / OT environment, which may include obtaining a new task distribution for the network, provisioning new deployments to affected devices, activating failover devices, or applying patches or updates.
    - Implement protection: The process of implementing and enforcing protection measures and mechanisms for an IoT / OT environment, such as device security, network security, or cloud security related to the change initiated.
    - Enterprise Posture Management: The process of managing and monitoring the security posture of an IoT / OT environment, including devices, services, vulnerabilities, threats, and incidents, and reporting the status and the results to relevant stakeholders and systems, such as users, cloud backends, or the intelligence function with alerts, secure logging or sharing a STIX threat report.
- Intelligence: The function of providing and consuming intelligence (information and insights) for an IoT / OT environment, using various methods and techniques, such as threat intelligence, vulnerability intelligence, or CI. The sub-functions of intelligence are:
    - Internal / External Threat Intelligence: The process of collecting, analyzing, and sharing threat information and indicators from internal or external sources, such as network traffic, devices, services, cloud backends, or third-party providers.
    - Vulnerabilities: The process of collecting, analyzing, and sharing vulnerability information and indicators from internal or external sources, such as network traffic, devices, services, cloud assets, backends, or third-party providers.
    - CI: The process of collecting, analyzing, and sharing intelligence (information and insights) from multiple sources and domains, such as IoT, OT and IT environments, including cloud environments.
    - Protection Engineering: The process of applying intelligence (information and insights) to the protection function, such as identifying and assessing the security posture, initiating and implementing changes and improvements, or implementing and enforcing protection measures and mechanisms.

The benefits to the CISSAN project's stakeholders include the acceleration of technology development and validation, improved interoperability, and the ability to make informed decisions

regarding the adoption of the technologies. At the European level, the project contributes to the enhancement of digital sovereignty, the resilience of critical infrastructures, and the development of advanced cybersecurity solutions aligned with the EU priorities.

2.2 High-Level Architecture

The CISSAN framework specifies the architectural principles, methods, and interfaces for implementing CI-based cybersecurity solutions developed in the CISSAN project. The CISSAN platform is the hardware and software implementation of the CISSAN framework that realise its architectural principles and mechanisms in practice as a system, including a CISSAN software framework that uses isomorphic software based on WebAssembly to ensure consistency across heterogeneous environments.

The CISSAN framework can be viewed (the upper part of Figure 1) as composed of four main layers: the IT / OT infrastructure layer, the data layer, the processing layer, and the domain / application layer. Each layer has a specific role and function in the framework and contains several components and subcomponents.

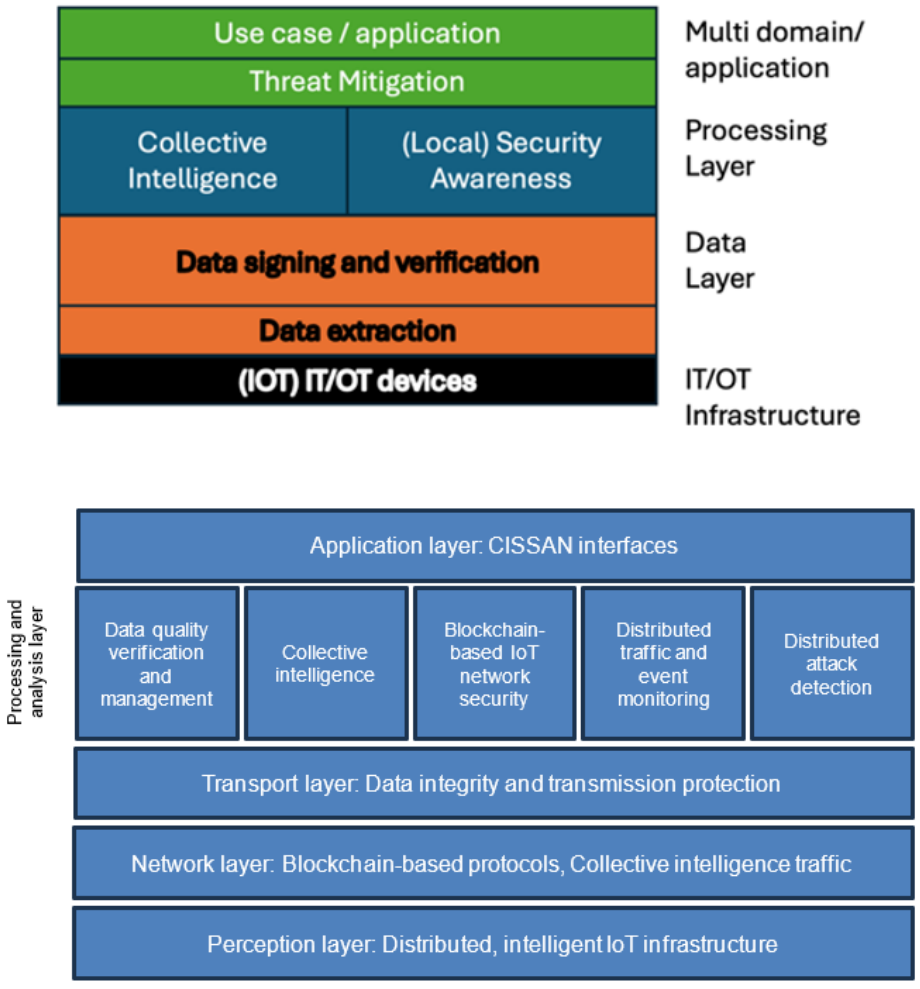

Figure 1. Two views of the layered architecture of the CISSAN framework

In another view (the lower part of Figure 1), aiming to provide a comprehensive and holistic view of the security requirements and challenges for IoT / OT environments, we define five main layers: Perception, Network, Transport, Processing and analysis, Application. Each layer represents a level of abstraction and granularity for the IoT / OT data and processing.

The layered architecture of the CISSAN platform, as shown in Figure 2, reflects a converged IoT/OT perspective and highlights its core components and their interactions within the overall system. It provides a clear modular integration, experimentation, and collaboration architecture for IoT/OT domains, thus reducing the risks of development and the likelihood of duplication of efforts among the CISSAN partners.

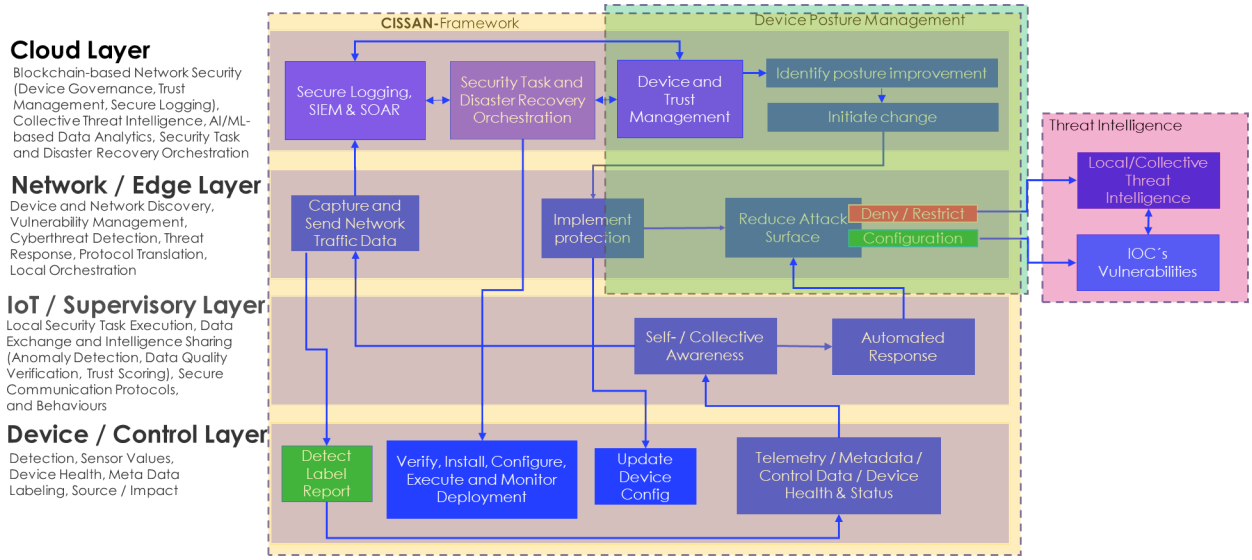

Figure 2. CISSAN platform layered architecture

The CISSAN platform is primarily deployed on-premise and consists of centralized components including a management server, a Supervisory Control and Data Acquisition (SCADA) server, a Security Information and Event Management (SIEM) and Security Orchestration, Automation, and Response (SOAR) server (referred to as the orchestration server), with the possibility of interaction with a blockchain network and cloud services and other external services provided by CISSAN partners, including optimization services, use case systems related services, for higher-level coordination, optimal security task allocation, secure data logging, data ingestion and analysis, which are represented by **the cloud layer**. In IoT and OT networks, the cloud layer refers to the collection of centralized or distributed computing services that offer scalable processing, storage, and management capabilities beyond those possible in the device or edge level. It is often the host of data aggregation, analytics, orchestration, security management, and application services. This allows for cross-domain visibility and coordination of numerous devices. In the case of OT networks, the cloud is often used in a controlled and segmented manner to offer monitoring, decision support, and lifecycle management functions while time-critical control functions are often closer to the edge or the on-premise systems. The cloud layer of the CISSAN platform performs blockchain-based network security, which includes device governance, trust management, secure logging, and data aggregation. It also aggregates and analyses collective threat intelligence, performs AI/ML-based data analytics, and security task and disaster recovery orchestration. Devices' posture including health statuses, device behaviour, security posture, configuration posture, identity and trust postures, are continuously monitored by the cloud. Appropriate actions are performed to enhance the device posture. This entails segregating the compromised or low-trust devices from the rest of the network, updating the security state of the devices in the device ledger, initiating remediation actions through the CISSAN orchestrator, and enforcing access control rules at the network level, including triggering device isolation at the switch level.

The **network / edge layer** is the communication infrastructure, gateways, and edge computing devices necessary to connect the IoT and OT devices to the platform and the higher-level services. This layer is essential for secure data exchange and processing and is critical for the enforcement of various security and access control decisions made by the orchestration and trust management functions. This layer is essential for responsive and resilient operation because it allows time-critical functions to operate close to the environment while still providing integration with the central monitoring and coordination functions. The network / edge layer in the CISSAN platform performs device and network discovery, vulnerability management, incidence detection, threat response, protocol translation, and local data and task orchestration.

The **IoT / supervisory layer** is composed of supervisory systems and data collection services. This comprises SCADA and telemetry ingestion components. It offers a consolidated view of the state of IoT and OT environments. It is used for monitoring, event and alarm management, and supervisory control. It offers the necessary data to the platform services for performing tasks like anomaly detection, trust scoring, and security orchestration. This makes it a significant integration point between the CI and security management functions of the platform. The IoT / supervisory layer in the CISSAN platform performs local security task execution, data exchange and intelligence sharing (anomaly detection, data quality verification, trust scoring), secure communication protocols, and behaviours.

The **device / control layer** consists of the field devices and control systems that interact directly with the environment, and these include sensors, actuators, controllers, and other embedded systems. This layer performs the task of data acquisition, control logic, and control of the environment, and it does so with several strict limitations, making it the foundation of the IoT and the OT system. The device / control layer in the CISSAN platform includes performing detection, sensor values, device health, meta data labelling, source / impact. Implementation of device security features include certificate-based device authentication, access control. In the device layer of IoT networks, devices sense and act on the physical world while running embedded logic, and send telemetry, metadata, and health/status information to the IoT layer, while receiving control commands and configuration in return. In OT networks, the control layer executes control of the physical process using sensor inputs and control logic, and sends process variables, states, alarms, and events to the supervisory layer for monitoring and higher-level control.

2.3 Design Principles and Constraints

2.3.1 CISSAN Security Principles

The CISSAN project is both (i) developing technologies and proposing methods to enable IoT and OT environments to follow this set of principles and (ii) following this set of principles in its own research and engineering efforts:

1. Security-by-design: Select and embed security and privacy features in target devices, products, systems, and platforms from the initial stages of the development lifecycle, not as an afterthought or add-on.
2. Least privilege: Grant the minimum level of access and permissions to devices and users, according to their roles and responsibilities, to reduce the potential impact of unauthorized or malicious actions.
3. Data minimization: Collect and store only the necessary data for intended purposes and delete or anonymize the data when no longer needed to protect the privacy of data subjects and to reduce the risk of data exposure.
4. Encryption: Encrypt data in transit and at rest, using strong and standardized encryption algorithms and protocols, to prevent unauthorized access or modification of the data.
5. Audit and accountability: Maintain logs and records of activities, events, and transactions in target devices, products, systems, and platforms and enable auditing and accountability mechanisms to monitor and verify the security and privacy of those.
6. Zero-trust: Enforce zero trust by continuously verifying identity, device integrity, and trust levels through a blockchain-based architecture that provides decentralised, tamper-resistant, and auditable security decisions in Internet of Things (IoT) and Operational Technology (OT) networks.

2.3.2 CISSAN Architectural Principles

The CISSAN architecture uses various techniques, such as deep learning, Generative Adversarial Networks (GAN), CI and blockchain, to enhance the security capabilities and performance of IoT and OT environments. CISSAN leverages the cloud and edge computing paradigms to enable efficient and scalable data processing and to employ cloud-based features, such as threat intelligence, advanced analytics, and cross-domain collaboration.

1. CISSAN leverages the CI of IoT and OT devices and backends to share security information and alerts, such as indicators of compromise, signatures, Structured Threat Information Expression (STIX) threat reports, or policies, and to coordinate planned responses and actions.
2. CISSAN adopts security-by-design and zero-trust approaches and implements security measures, including network segregation, trust management, access control, security monitoring and logging, intrusion detection and prevention, data privacy protection, physical security, and incident response including automated disaster recovery, device blacklisting and device isolation.
3. CISSAN implements monitoring tools and processes to continuously monitor the health and performance of IoT and OT platforms and devices, including relevant data flows monitoring.
4. CISSAN creates a set of security management and governance methods and documents, comprising processes, policies, standards, and best practices, to guide the design, implementation, integration, and operation of IoT and OT platforms and devices and to help achieve the compliance with relevant regulation requirements.
5. CISSAN uses cryptographic methods to sign and verify the data exchanged among endpoint and edge devices and cloud backends, ensuring the authenticity and integrity of the data.
6. CISSAN covers various connectivity technologies, such as Wi-Fi, Bluetooth, Zigbee, LoRaWAN, cellular and other, to ensure communication security. The project utilizes gateways to aggregate data from IoT and OT devices, translate protocols, perform edge computing, and make local decisions. Gateways may implement other security functions such as device authentication, data encryption, and access control.
7. CISSAN anticipates future growth, new devices, technologies, and changing business requirements by designing a scalable and flexible IoT / OT security framework. The project follows a data-driven approach and implements data management processes such as data collection, storage, ingestion, processing, validation, analysis, visualization, security, and governance. It also supports task and data orchestration, and blockchain-based device governance.
8. CISSAN ensures seamless integration of its security mechanisms with existing enterprise systems, third-party services, and APIs for data exchange, business process automation, and decision-making.

3 CISSAN Platform Logical Architecture

3.1 Core CISSAN Platform Components

This section defines the actual system components within the CISSAN platform deployment. Rather than listing every individual host, assets are grouped by function and operational role. This provides a consistent reference model for describing where platform components are deployed, how they connect to the use case system, and where security and collective-intelligence mechanisms operate.

The CISSAN platform assets are grouped by function and role in the network. The environment is intentionally multi-domain and combines industrial control components, IoT/edge devices, experimental orchestration services, and the network/security infrastructure that binds them. Virtualized systems are not treated as a separate class: VMs and containers are analysed within their operational segment (e.g., a virtual SCADA server belongs to the ICS category). Categorization is driven by (i) primary function, (ii) typical communication planes (control, telemetry, management, orchestration, remote access), and (iii) the asset's impact on confidentiality, integrity, availability, isolation, and accountability.

Figure 3 illustrates the CISSAN platform components (see implementation from CISSAN D6.1 CISSAN platform and solutions for the project use cases). The physical and logical core of the CISSAN platform consists of a 5G-enabled gateway router providing Wide Area Network (WAN) connectivity and secure remote access, a managed Layer 2 switch implementing VLAN isolation, and a Hyper-V virtualization host supporting SCADA systems, engineering virtual machines, and research workloads. Authentication is centralized via Remote Authentication Dial-In User Service (RADIUS) services, and secure remote connectivity is provided through VPN tunnelling. Continuous monitoring, event logging, and performance analysis mechanisms ensure visibility across network, device, and security layers.

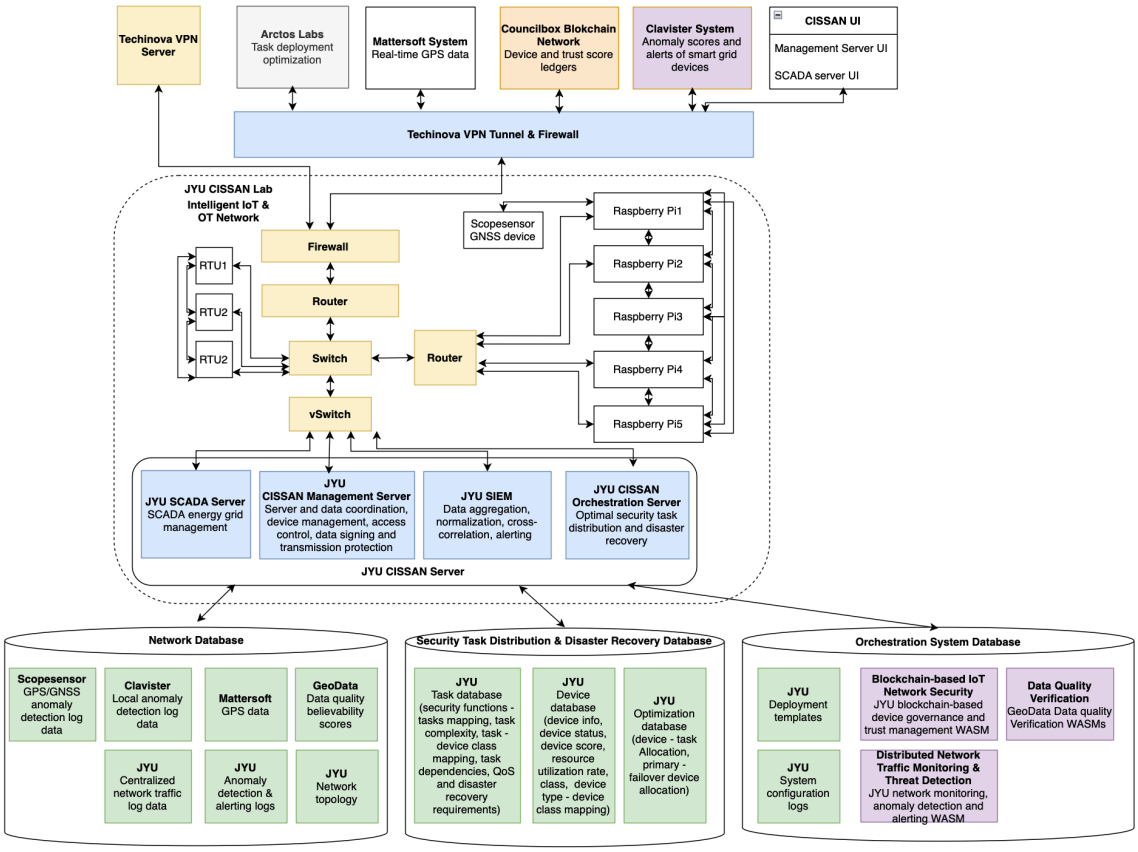

Figure 3. CISSAN platform components

Industrial Control Layer (ICS) — Field and Control

This category comprises Remote Terminal Units (RTUs) / Programmable Logic Controllers (PLCs), the SCADA server, and engineering workstations that implement and supervise industrial processes. These assets execute safety- and mission-critical control flows, so integrity and availability dominate other properties. Treating them as a distinct category isolates the control-plane from less-trusted domains and allows deterministic traffic patterns, peer allow-lists, and fail-safe behaviour without conflating them with general information technology (IT) services. Representative assets include 3 RTU units, SCADA server hosted on a windows virtual machine (VM), and ICS engineering workstations hosted on a Microsoft hypervisor, Hyper-V, when used for control engineering. Hyper-V is a native, type-1 hypervisor that can be used to create and manage VMs to virtualize applications, improve security, and increase hardware utilization. The main features of Hyper-V are virtual switching, isolated VLANs, and nested virtualization to build complex IT environments.

IoT and Edge Computing

In the CISSAN platform and use case systems, IoT devices collect, process, and exchange data over the Internet and other communication networks. In the CISSAN platform, IoT devices includes raspberry pi hosts, lightweight sensors and actuators, and embedded ML workloads at the edge. These nodes are numerous, heterogeneous, and often resource-constrained, which increases configuration drift and exposure to commodity attacks. Modelling them separately reflects their role as common entry points and acknowledges that feasible controls differ from those used for ICS or server assets. The key security properties here relate to segmentation, user access policies and the handling of data. Representative assets include five Raspberry Pis used for IoT and ML experiments.

Security Messaging Overlay using Message Queuing Telemetry Transport (MQTT)

This category captures the pub/sub overlay used primarily for security telemetry and coordination and optionally for other experimental data flows. Pub/sub overlay is a logical network infrastructure constructed on a physical network, for example, the internet or a P2P network and is used for enabling asynchronous communication between the publisher and the subscriber. It helps in routing the message in a distributed, scalable and efficient manner, and ensures high availability and low latency in IoT or distributed systems. Keeping the pub/sub overlay separate makes broker-level guarantees such as authentication, authorization, per-topic isolation, TLS, and bridging policy explicit, since they govern how information propagates across otherwise segmented zones. Representative assets and services include the MQTT broker, which provides the security messaging overlay, on a 5G router and MQTT clients on three RTUs, the SIEM server, five Raspberry Pis, and, where required, the CISSAN management server. Co-locating the broker on the 5G router couples the messaging overlay with the perimeter/edge router, which provides the remote access and network infrastructure. These are treated as two logical components. The edge gateway and the MQTT broker are considered shared-fate components, and the associated risk is explicitly identified and documented in the CISSAN platform security analysis report, which is provided as the confidential annex of the deliverable D2.2 report.

Application and Orchestration Services

This category comprises components responsible for security task coordination, workload scheduling, and distributed deployment logic within the CISSAN platform. These services determine where and when WebAssembly (WASM) modules are deployed and executed, and how optimization logic assigns tasks to edge devices. Because orchestration influences compute placement and data paths across zones, misconfiguration or compromise can indirectly impact multiple segments simultaneously. Representative assets include the CISSAN orchestration server, the optimization solver service, and associated APIs, as well as WASM modules deployed on IoT devices under orchestration control.

Management and Monitoring

This category includes components that provide centralized configuration authority, identity services, logging, and operational visibility across the lab. Compromise of the management plane typically yields broad and often silent control over multiple zones. Conversely, strong identity enforcement, role separation, and tamper-evident logging increase overall system assurance. Representative

assets include the CISSAN management server, the SIEM platform, blockchain components supporting trust verification, and the AAA services (FreeRADIUS) used for VPN and administrative authentication. These systems anchor identity, policy enforcement, and security telemetry across the deployment. The main virtualization host also belongs within the category for its critical role in maintaining the various virtualized assets.

Network and Security Infrastructure

This category comprises the switching, routing, firewalling, and foundational services that enforce segmentation and traffic policy between zones. These components act as isolation primitives and determine which communication planes are reachable across VLAN boundaries. Because all inter-zone traffic traverses this layer, configuration errors or device compromise can invalidate higher-level security assumptions. Representative assets include the core switch, the 5G edge router and firewall, and supporting routing devices, together with foundational services including DNS, DHCP, and NTP where provided.

Remote Access and Edge Gateways

This category covers VPN endpoints and border services that define the effective perimeter for remote administrators and researchers. Endpoint posture, tunnel authentication, and exposure management directly shape the external attack surface. Representative assets include the WireGuard VPN gateway on OpenWrt virtual machine and the 5G modem providing the external VPN link for accessing the lab.

Collaborative and Multi-Tenant Zone

This category encompasses shared development and research workloads operated jointly by project partners. It is intentionally lower-trust and characterized by higher configuration churn and experimental deployments. Strong containment, egress control, and tenancy isolation are therefore central to maintaining overall platform integrity. Representative assets include development systems, collective intelligence testing systems, and the patch panel workstation used for RTU-focused security testing.

IoT and OT devices often have limited resources to run security functions, or their vendors simply do not make it possible to integrate security functions to such devices. To improve security in such cases, one can add IoT / OT environment security sensors that can be used to analyse network traffic from other devices, copying or intercepting it, and to label and extract metadata for further use, e.g., in AI / ML-based cybersecurity solutions. We note that ML models can be trained and do inference in clouds, endpoints or edge (e.g., gateway devices). There are multiple options of training, including local training, aggregation of local data in a cloud, aggregation of locally trained models in a cloud, or aggregation of locally trained models in a group of devices. While inference is usually done either in a cloud or locally, the results of local inference in multiple devices can be combined / aggregated further or inference tasks can be distributed among multiple devices or delegated to other devices1. Various forms of aggregation, distribution and delegation of training and inference tasks (and more broadly other security tasks) can be considered CI, and identifying and implementing forms suitable for relevant use cases (including the project use cases) is on the CISSAN agenda. CISSAN is exploring the possibility of using the security functionality of IoT / OT nodes jointly with security sensors, for example, to detect or request blocking of peer-to-peer traffic between nodes that indicates malicious activities. In addition to ML-related tasks, reporting and sharing threat information among IoT / OT nodes is also a form of CI supporting security awareness in IoT / OT environments. Nodes and security sensors can be used to initiate and implement protection and mitigation measures, such as blocking, isolating, or device patching, to reduce the attack surface and prevent further damage.

An example of a security sensor is network tap, which is a device that captures the network traffic from IoT / OT devices and sends it for analysis. It can be either a physical device installed in a network infrastructure, e.g., a switch or a router, or a piece of software installed in an IoT / OT or other device as a virtual machine or a container. A network tap can be either a passive device that only copies the network traffic and sends it for analysis or an active device that can also intercept and modify the network traffic, such as a firewall or a proxy. It can capture the network traffic from either a single IoT / OT device or multiple devices, e.g., from a network segment or a subnet.

Edge is a distributed and local computing platform that provides certain services and resources for IoT and OT environments, including data processing, data analysis, and data exchange. CISSAN can benefit from such edge-based features (parts of the framework) as Asset Discovery, Vulnerability Management, Cyberthreat Detection, Threat Response, Data Exchange, Data Filtering, Data Aggregation, and Data Compression. Examples of edge device types are gateway, router, and switch. Edge devices can also be a target or a vector of cyberattacks, and therefore they need certain security features, such as access control, encryption, or firewall.

Cloud can be used to support leveraging AI and ML techniques to enhance the security of IoT and OT environments. AI and ML can be instrumental in analysing large and complex data, detecting and classifying known and unknown cyberattacks, providing situational awareness and risk assessment, and automating and optimizing attack response and mitigation. Obviously, cloud environments can also be a target or a vector of cyberattacks, which must be considered in threat analysis, implementation, integration, and operation.

Depending on the device capabilities and the use case domain, cybersecurity functions at the device level can include certificate-based authentication, remote attestation, secure boot, and secure protocols to prevent data theft, tampering and spoofing. For instance, an IoT sensor is a specific type of device (or part of a device) that detects events or changes in its environment, sending collected data to an IoT gateway, other edge devices, other IoT devices, or cloud backends. IoT sensors typically directly interact with the physical world. For example, a temperature sensor in a smart thermostat collects data about the room temperature and sends it to the system to adjust heating. IoT node refers more broadly to any physical device within an IoT system, which includes sensors but also other components such as actuators, cameras, and gateways. An IoT node can be as simple as a sensor or as complex as a gateway that aggregates data from multiple sensors and performs certain data processing before sending it to cloud backend or other system.

3.2 Use Case 1 (Transportation) — System Components and Roles

In Use Case 1 (UC1), the CISSAN platform is integrated with a public transport vehicle telemetry environment where positioning information is derived from GPS/GNSS satellite signals and made available for collective anomaly detection. The use case is designed as a non-intrusive integration where CISSAN consumes positioning data for analysis and does not provide any operational control path to vehicles.

Vehicle-side (Public Transport Fleet)

The public transport fleet consists of operational vehicles such as buses and trams. Each vehicle contains a GNSS/GPS receiver that computes positioning fixes based on satellite signals. The GNSS receiver is typically integrated into a vehicle telematics subsystem, where a driver terminal (on-board computer) and associated software collect and process positioning data for operational use (e.g., vehicle monitoring, passenger information, and dispatch). Vehicles are connected to backend systems through a mobile gateway (cellular modem/router), typically operating via an APN-based connection to provide controlled connectivity to the operator environment. The vehicle-side system is considered an OT/IoT-like environment in the sense that it consists of embedded devices with constrained interfaces, and its primary function is operational continuity and data integrity.

Backend and Data Transport

Vehicle positioning messages are transmitted from the fleet to backend servers operated in the transport telemetry environment. These servers aggregate vehicle location messages and forward them through a secure integration pipeline. The internal details of these upstream backend components are outside the scope of the CISSAN laboratory platform and are treated as an external data source for the purpose of the project.

CISSAN Lab Edge and Messaging Infrastructure

In the CISSAN laboratory environment, the primary integration point for UC1 is an MQTT broker that receives vehicle positioning messages from the upstream telemetry environment. The MQTT broker functions as a secure pub/sub data distribution component and is the boundary where the use case

data becomes available to CISSAN components. The integration is implemented in a read-only manner from the CISSAN perspective: CISSAN subscribes to defined topics to consume positioning messages for analysis.

IoT / Edge Compute Devices (Raspberry Pi)

In UC1, Raspberry Pi devices represent the IoT/edge computing layer of the CISSAN platform. These devices are used to execute anomaly detection and related analytics tasks over the incoming vehicle positioning streams. The Raspberry Pi nodes host lightweight software modules that perform data processing and anomaly scoring, enabling distributed execution and experimentation within the CISSAN platform. The Raspberry Pis are not part of the vehicle system; they are CISSAN lab components used to demonstrate how CI can be applied to positioning telemetry at the edge.

Servers and Central Platform Components

The CISSAN platform includes centralized services such as the management server, logging/SIEM components, and orchestration services. In UC1, these components provide coordination, monitoring, and aggregation of results produced by the edge analytics layer. The management and monitoring layer provides visibility into system operation and supports the demonstration of CI workflows, such as the propagation of anomaly evidence, storage of analysis outputs, and optional integration into trust scoring mechanisms.

Network Security Components

Network security is provided through the lab's routing and firewall infrastructure, which enforces segmentation and controlled connectivity between the external data source, the MQTT broker, and the internal CISSAN platform components. This supports the security-by-design and least-privilege principles by ensuring that the UC1 integration remains limited to the intended telemetry ingestion and does not introduce a control-plane exposure toward the transport fleet.

3.3 Use Case 2 (Smart Grids) — System Components and Roles

Use case 2 (UC2) validates the CISSAN platform in a smart-grid style OT setting centred on an ICS zone that simulates supervisory control and field-device operation. The deployment is built around an operational control loop and a parallel security messaging overlay that enables collective-intelligence functions such as, telemetry sharing, comparison, trust exchange, and coordination of response actions without interfering with the deterministic control-plane. The following components constitute the core of the use case system and define its operational and security roles.

Supervisory Layer (SCADA)

The supervisory layer is implemented by a Zenon SCADA server hosted as a virtualized system within the lab's compute environment. The SCADA server provides supervisory control and RTU communications to emulate real operating conditions in a smart-grid context. SCADA–RTU interaction in UC2 is treated as control-plane traffic, since its integrity and availability define the realism and stability of the operational process being simulated. From the platform perspective, the SCADA server therefore serves both as an operational anchor for the use case environment and as a reference point for assessing the applicability of the developed security solutions against OT workflows.

Field and Control (RTUs)

The field layer comprises three RTU units that represent remote terminal devices in a smart-grid environment. These RTUs implement the operational endpoints of the simulated process and are the primary targets for the UC2 applied security mechanisms. Operationally, they receive supervisory interactions from SCADA and provide corresponding telemetry. From a security perspective, UC2

uses the RTUs to demonstrate distributed defence: each RTU participates in local observation, produces lightweight evidence of its own state, exchanges trust-related messages with peers, and contributes to a collective understanding of device trustworthiness. The RTUs connect to the security messaging overlay to share security telemetry and coordination signals and to support use case actions such as trust scoring and automated mitigation of threats (isolation of compromised units).

Security Messaging Overlay (MQTT)

The CISSAN security overlay for UC2 is implemented using MQTT as a lightweight, decoupled communication fabric. MQTT is used to carry inter-device trust messaging (RTU-to-RTU and RTU-to-management), resource and status reporting from participating nodes, and coordination signals such as policy propagation or response triggers where applicable. The pub/sub model supports asynchronous communication and enables multi-producer/multi-consumer telemetry distribution without requiring modifications to OT control protocols. In UC2, the overlay is treated as a security telemetry plane that complements the control plane: it enables collective-intelligence functions to aggregate and contextualize evidence across nodes while keeping the operational control loop separate. Broker access is assumed to be constrained to authorized lab connectivity and is not intended to introduce a public Internet exposure.

Management and Monitoring

Central platform services provide coordination, aggregation, and visibility for the UC2 security functions. Telemetry and trust evidence published via the MQTT overlay are mirrored to the monitoring stack for correlation and investigation, and security-relevant events are surfaced through the SIEM interface. In parallel, a management GUI provides an operational overview of device status and the evolving trust state used in the demonstration. This separation of responsibilities mirrors common practice: the SIEM supports auditability and post-event analysis through log-centric workflows, while the management interface provides real-time visibility into collective-intelligence state and orchestrated response actions at system level.

Network and Security Infrastructure

Network security for UC2 is provided by the lab's segmentation and traffic policy enforcement, which constrain how control-plane and telemetry-plane communications traverse zones. This infrastructure enforces the boundary conditions required by the use case: the SCADA-to-RTU control loop remains confined to the ICS segment, while the MQTT overlay and management-plane access are reachable only along explicitly permitted paths. This supports least-privilege design by limiting lateral movement opportunities and by ensuring that security coordination mechanisms do not unintentionally create new routable access into the OT control plane.

External Assets

In addition to the core lab ICS components, UC2 includes partner-hosted assets used to extend the demonstration context beyond a single local segment. These external RTU instances are used to demonstrate novel anomaly detection solutions and related security workflows, within partner environments. However, even these solutions are integrated into the platform through agreed messaging and API interfaces, enabling the outputs of different solutions enforce cross-environment/solution visibility while keeping operational control local to each environment.

3.4 Use Case 3 (Tunnel Construction) – System Components and Roles

In use case 3 tunnel construction, the devices used include:

- Tunnel monitoring sensors, e.g. measuring tunnel deformation,
- Infineon security chips signing the measuring data at creation/at the sensors,

- a gateway located at the tunnel portal to which the sensor data and data signatures are transferred and where the signatures are verified the first time, and from which sensor data and signatures are further transmitted to
- multiple nodes (implemented in form of single board PCs) of the Lightning Network, a "Layer 2" payment protocol built on top of the Bitcoin blockchain. Through the network the data is further transmitted to the
- cloud-based data management server GeodataHub where the data signatures are checked again, the data quality verified and the data finally stored and provided to the end users for decision making.

IoT and Edge devices

For use case 3, the Raspberry Pi devices are used to represent the sensors, gateways and lightweight compute nodes of the tunnel construction use case. They are used to execute the WASMs, i.e. the believability scoring and trust scoring modules, which are used to illustrate the use of collective intelligence mechanisms with resource constrained devices on the CISSAN platform.

Orchestration and Optimization

The distribution of the WASM deployments is coordinated by the CISSAN Orchestrator residing on the orchestration server and the optimization solver tool. The optimization solver selects the best Raspberry Pis for the WASMs based on their available resources and their trust level. In cases where a device that is part of a deployment, has a problem, the CISSAN Orchestrator and optimizer tool will recompute the deployment and the WASMs will be redeployed.

Management and monitoring

Believability and trust scores are stored at the management server which also provides view of the devices of the network. Information displayed includes device trust levels and blacklist status, anomaly/believability scores, WASM placement and system health. The management server communicates with the blockchain to store device blacklist status and trust scores.

Network Security

Network security is handled with the lab’s routing, firewall infrastructure and access controls, which enforce segmentation between the internal CISSAN platform components, supports the security-by-design and least-privilege principles.

3.5 Use Case 4 (Bittium Manufacturing Execution System) – System Components and Roles

Netox-led cybersecurity architecture design for CISSAN is validated and tested it in the Bittium BMES Use Case. The objective was to realize *security-aware nodes* and distributed detection across IoT/OT, edge, and IT platform layers, leveraging *Microsoft Defender for IoT* sensor for passive network telemetry with Sentinel as the SIEM plane and SOC Radar for external Threat Feeds. These choices follow the current architecture picture circulated for Section 2.2 and the guidance to focus on concrete components and interfaces. The implementation is defined through Bittium use case.

Bittium BMES is a containerized, microservices-based MES platform that Bittium uses for lifecycle management of manufactured medical devices. The system is designed to scale to hundreds of containers (e.g., one per manufacturing partner); business services are exposed as REST APIs; the UI layer is React; data back ends include MS SQL Server and PostgreSQL. For architectural design, development, and validation, Bittium provided a large device dataset combining generated and non-classified real-world test data.

Implementation

Following the CISSAN architecture, Netox designed positions of Defender for IoT sensors at Bittium BMES network vantage points NGINX (SPAN/TAP) to perform asset discovery, vulnerability exposure mapping, and threat detection on east-west and north-south traffic, forwarding normalized telemetry and alerts to the cloud management plane and Microsoft Sentinel for correlation, hunting, and automated response. The architecture also anticipates certificate-based device authentication, remote attestation, and secure boot/protocol primitives in the device/edge layer where available via BMES capabilities.

Distributed System Elements and Interfaces

Device & Edge layer.

- *IoT/OT devices*: BMES use-case endpoints (manufacturing-related device fleet represented by the Bittium dataset) and site infrastructure components that generate operational telemetry. (Dataset-based representation for development/validation.)
- *Gateways/edge nodes*: Network elements that forward traffic; vantage points for passive sensors.
- *Security sensors*: Defender for IoT on-premises sensors (virtual appliance form factor) receiving mirrored traffic; sensors extract metadata/features locally and forward findings to the cloud management plane.

Bittium BMES application layer.

- *Microservices*: Containerized (Docker) services (hundreds at scale) providing business capabilities via REST endpoints.
- *Data services*: MS SQL Server for relational back-office/state; PostgreSQL for operative stores.
- *UI layer*: React front ends invoking service APIs.

Netox security & management layer.

- Defender for IoT Cloud, Management of sensors, Threat analytics, CTI feeds, and ML/AI-enhanced detections. Hunting querys-
- Microsoft Sentinel for alert ingestion, correlation, case management, and automation playbooks.
- Biitum device control for secure device-to-control communications where the use case requires command/control alongside passive monitoring.

Device, Edge, and Agent Layers

- Device layer: Sensors/actuators in the manufacturing lifecycle; modeled by Bittium's 1,000-device dataset for development and test.
- Edge layer: Gateways and virtual switches enabling SPAN/TAP; Defender for IoT sensors run as virtual appliances adjacent to BMES traffic paths.
- Agent layer: (Where applicable) lightweight agents for endpoint/host telemetry complement network-centric detection, with events fused in Sentinel.

The above componentization conforms with CISSAN's secure platform vision, and the security-aware nodes rationale as introduced in the project description documents.

Interfaces, integrations and protocols

BMES is a microservices-based system (React front end, REST services, MS SQL Server & PostgreSQL back ends) designed to scale to hundreds of containers—often one per manufacturing partner. Interfaces and protocols are the contract that keeps this scale from devolving into brittle point-to-point coupling.

Netox's stack relies on Defender for IoT sensors (passive network visibility) feeding Sentinel (SIEM) and D3 (SOAR). Unless you pin down the exact feeds, schemas, auth patterns, and network placements, detections and automated response cannot be guaranteed or audited. Also considering human in the loop (HITL) in BMES use case is important.

Internal Service Interfaces.

- *BMES Service APIs*: REST endpoints consumed by the React UI and partner services. (Security: OAuth2/OpenID Connect recommended; API gateway logging to SIEM.)
- *Data access*: App-to-DB over segmented virtual networks to MS SQL Server/PostgreSQL. (DB audit to SIEM.)

IoT and OT Device Interfaces.

- Device telemetry/control via site networks and gateways; Bittium “IoT Hub” is available for secure device-to-cloud messaging when used; device trust reinforced by certificate-based authentication, remote attestation, and secure boot.

Systems Integration (BMES ↔NETOX)

- *Network telemetry*: SPAN/TAP → Defender for IoT sensors (virtual appliances). Sensor-to-cloud uses secure outbound connection; incidents stream to Sentinel.
- *SIEM/SOAR*: ingests Defender for IoT alerts plus BMES logs (API gateway, container runtime, orchestration, DB). Playbooks orchestrate containment (e.g., network ACL updates, service isolation).
- *Threat Intelligence*: Cloud-based TI and ML/AI analytics augment detections and with *supervised learning* logic.

Data Exchange Formats and Protocols.

- *Traffic capture*: PCAP (SPAN/TAP) for sensor ingestion.
- *Security events*: Defender for IoT → Sentinel via native connector (JSON).
- *Application logs*: JSON/CEF common schema to Sentinel

The design directly implements CISSAN’s security-aware nodes concept—distributing detection and response across nodes and layers—and supports collective intelligence by sharing local insights (sensor analytics) into a central reasoning plane for fused, cross-node decisions.

Distributed Network Traffic Monitoring & Threat Detection

Passive, out-of-band network monitoring using Defender for IoT sensors provides comprehensive L2–L7 visibility without requiring intrusive instrumentation inside BMES containers. Sensors extract metadata and behavioral features locally, enabling both local anomaly detection and system-level analytics through the CISSAN plane.

Sensor placement

- Primary taps on BMES container fabric ingress/egress (north-south) and service-mesh/inter-service segments (east-west) using SPAN/TAP on the virtualized host switches or container network overlays.
- Optional taps at partner-link edges where BMES instances per partner terminate.
- Where encrypted traffic limits inspection, rely on metadata (SNI, JA3/JA4, flow behavior) and control-plane logs for context; use selective decryption only in zones approved by Bittium security.

Telemetry pipeline.

1. PCAP/SPAN mirrors feed Defender for IoT sensors.
2. Sensors perform asset discovery, vulnerability posture inference, protocol parsing, and threat analytics, and export findings to Defender cloud; enriched incidents stream to Microsoft Sentinel for correlation with identity, endpoint, and *cloud signals.*
3. BMES service and infrastructure logs (API gateway, container runtime, DB audit) are forwarded to Sentinel via connectors for fused detections and playbooks.

Security Management

- Defender for IoT deployment: Provision virtual sensors sized to BMES throughput; register with Defender portal; define zones aligned to BMES network segments; enable cloud analytics and TI feeds.
- Sensor connectivity: Outbound-only management plane; no inbound exposure required. (Meets BMES security constraints.)

Orchestration & Automated Response (presented method)

- Implement analytics rules tuned to BMES protocols and flows; automate triage/containment (e.g., block indicators, isolate services, open incident with context to BMES operators).

- Collective intelligence signals (e.g., distributed anomaly scores) are shared to reinforce detections across nodes.

3.6 Use Case 5 (Joint Use Case) – System Components and Roles

In use case 5 – joint use case, the devices used include:

IoT / Edge Devices

The five Raspberry Pi devices from the IoT zone represent sensors, gateways, and lightweight compute nodes originating from the tunnel-construction scenario. In UC5 they execute WebAssembly (WASM) modules such as anomaly detection, believability scoring, and trust evaluation. These edge nodes publish telemetry and trust evidence to the MQTT overlay and receive orchestration instructions for task placement. Their behaviour illustrates how collective intelligence mechanisms operate in resource-constrained IoT settings and how trust scores influence task assignment and device participation.

Orchestration and Optimization Services

The CISSAN Orchestration server and optimization solver coordinate the deployment and migration of WASM modules across Raspberry Pis. In UC5, orchestration is driven by device trustworthiness: devices with sufficient trust scores receive workloads, while devices with declining trust are excluded. When a node becomes unreachable or loses trust, the optimizer recomputes the deployment, and the CISSAN Orchestrator shifts tasks to alternative nodes. These services form the backbone of the automated recovery demonstrated in UC5.

Management, Monitoring, and SIEM

The CISSAN management server provides a unified view across UC1, UC2 and UC3, displaying device trust states, module placement, and system health. The SIEM receives telemetry and trust-related alerts from both domains, enabling correlation and cross-domain threat understanding. In UC5, this layer demonstrates how system-level situational awareness is formed when heterogeneous domains share evidence and collectively respond to anomalies.

Councilbox Blockchain System

The Councilbox Blockchain System (Eventchain) is a fully custom blockchain written in Python, deployed alongside the CISSAN platform as an external system for Use Case 5. It is not based on any existing blockchain framework (Ethereum, Hyperledger, etc.) and provides the trust management and device governance infrastructure for the use case. The system consists of a master node that creates blocks and coordinates 51% BFT consensus, hub nodes that collect IoT events, validate blocks, and maintain local blockchain copies with deterministic anomaly detection models, and a dual RabbitMQ broker architecture for secure inter-node communication. The CISSAN management server interacts with Eventchain through the Metadata API, a stateless REST service that records device registrations, trust score updates, anomaly events, and blacklisting decisions as consensus-validated blockchain transactions. A web-based Blockchain Explorer provides read-only audit and inspection of the ledger.

3.6.1 Management

The CISSAN Management Server is the central platform component that coordinates the CISSAN security and data-quality workflows for the field layer. It runs as a headless daemon with no HTTP API: it integrates with the CISSAN Orchestrator for device, module, and deployment lifecycle; subscribes to the MQTT overlay for RTU and device telemetry and trust-related messages; and maintains a device registry backed by a static port-isolation allowlist so that only authorised nodes participate in collective-intelligence processing. Operationally, it drives a control-plane loop that builds optimization requests from the current network and task inventory, calls the optimization solver, and creates and activates deployments via the CISSAN Orchestrator so that WASM workloads (including trust scoring and anomaly detection) are placed and run on the intended edge devices. A separate data-plane loop periodically triggers task execution on device supervisors over HTTP, retrieves and parses module outputs (e.g. CSV-based scores), and aggregates per-device trust,

believability, and anomaly results. This aggregated state is published to MQTT as retained topics, providing a single, consistent view of device and trust state for the management GUI and other consumers. From a security and response perspective, the management server enforces trust-based policy: when a device's trust score falls below a configured threshold, it triggers port isolation via the blacklister API, updates blacklist status in the CISSAN Orchestrator and optionally on the blockchain and publishes blacklist and recovery events over MQTT. The management GUI, which consumes these MQTT topics, gives operators real-time visibility into device status, trust evolution, optimization and deployment state, and orchestrated response actions, while telemetry and events can be mirrored to the monitoring stack for SIEM correlation and audit. In this way, the management server acts as the coordination point that ties together the MQTT overlay, orchestration, edge execution, and trust enforcement into a single platform view.

3.6.2 Data Quality Verification

The purpose of the Data Quality Verification module is to calculate the local believability scores on IoT devices, which are used in obtaining local trust scores of nodes, which are then aggregated on the CISSAN management server for calculating global trust scores of the networked devices in the CISSAN platform. The module also calculates an overall believability score to validate the results. If a discrepancy is identified, the module triggers anomaly detection to the "Distributed Monitoring and Threat Detection" module. If the believability score of data is identified as low (below a pre-defined threshold), anomaly detection is performed on the data at the CISSAN server using the aggregated data obtained from the SIEM. We adopt variance-based and similarity-based methods in data quality verification. The variance-based method is able to detect anomalies using local linear regression within a sliding window. The similarity-based method uses dynamic time warping (DTW) to measure similarity between two time series (e.g., from neighbouring sensors). More details on the methods can be found in CISSAN deliverable report D5.4.

In the tunnel construction use case, geotechnical sensor readings (e.g. elevations of monitoring targets) are manipulated to generate false safety readings. The consequence is that an unstable tunnel goes unnoticed. Data transmitted to engineers are falsified. Such incidents can be detected in the CISSAN application server as shown in Figure 4.

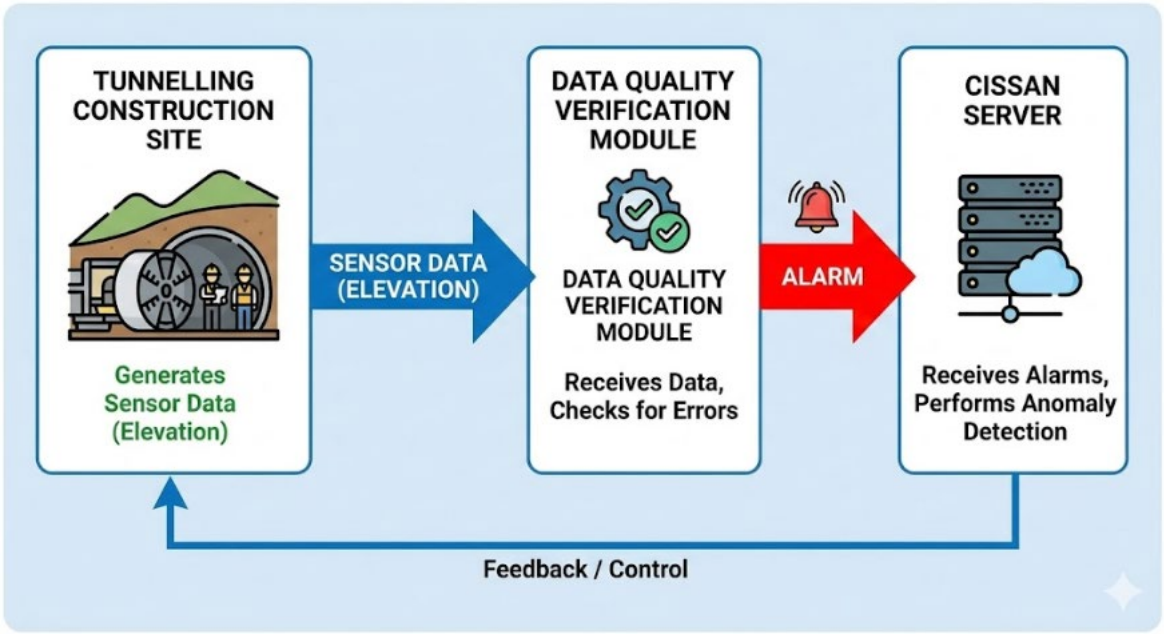

Figure 4. Data quality verification

3.6.3 Distributed Network Traffic Monitoring and Threat Detection

Distributed network traffic monitoring and threat detection in Rotor

This section relies on concepts explained in CISSAN deliverable report D5.4 regarding CI and the Rotor framework, as well as in D5.2 regarding local monitoring mechanisms.

Distributed threat detection and network monitoring in the lab was implemented by running lightweight monitoring directly on OT devices, enabling detection from within the operational environment rather than relying solely on perimeter controls or centralized traffic inspection. Each device performs local security monitoring and produces structured anomaly observations that are then shared based on CI logic for peer validation and central aggregation. The monitoring logic is designed to remain operationally feasible on constrained industrial hardware through resource analysis, minimal design and by combining specification-based checks with deviation-oriented heuristics, rather than heavyweight analytics. For distributed network traffic monitoring in Rotor, the design intentionally avoids continuous packet inspection and instead relies on lightweight, periodic snapshots suitable for resource-constrained OT devices, as part of the local monitoring modules. The low-level monitoring, is built as a modular Rust program, which was then compiled into a standalone binary for the target environment, with no external dependencies. The binary is managed by python-based services, which ensure timely collaboration, execution and standardised response to incoming peer information. Distribution of these signals is established via MQTT publish-subscribe architecture, facilitated by a broker located within the OT network, on the 5G router. The broker is responsible for enabling inter-device trust and security information messaging. In practise: RTU-to-RTU and RTU-to-Management.

3.6.4 Trust, Device and Identity Management

Trust Management in OT network

In the OT laboratory network, trust management was implemented as a device- and peer-oriented mechanism that translates observed consistency into operational confidence. The target devices are the RTUs, and the key supporting infrastructure for this effort consists of a 5G router for CI overlay, L2 switch for VLAN trunking and segmentation, and the CISSAN management server for trust management.

Devices exchange structured security observations over the MQTT based CI overlay and compare received reports against a locally generated snapshot, allowing each node to form an independent view of whether a peer's behaviour aligns with expected conditions. This comparison is performed by a peer comparison and evaluation service, implemented in Python for portability on constrained RTUs, which validates incoming messages, avoids self-feedback, and persists received data for transparency and auditability. The comparison evaluates the alignment of reported anomaly categories producing a quantitative consistency value that captures how similarly two devices observe security-relevant behaviour under comparable operating conditions.

Trust is updated based on these outcomes and propagated as compact trust deltas to a central management component that maintains global trust state across the network. At a conceptual level, the trust model reflects three mechanisms:

- Trust increases based on sustained agreement between peer observations.
- Trust decreases based on sustained disagreement, indicating potential compromise, misconfiguration, or drift.
- Trust decreases due to failure to cooperate (e.g., missing reports or inactivity).

Technically, trust updates follow an exponential moving average, configured so that consistency accumulates trust gradually while deviations can reduce trust rapidly, providing responsiveness without overreacting to transient noise. Where needed, the mechanism can also be extended to weight updates by the severity of disagreements, enabling sharper differentiation between minor deviations and high-impact anomalous behaviour. Failure to cooperate is visible in the system, however, the mechanism to decrease trust in this regard, is currently not deployed (See D5.4 for more detailed information).

Trust management of the IoT devices

Trust management of the IoT devices on the CISSAN platform is represented by the Raspberry Pi devices and is executed via the TrustScoring WASM module deployed with the CISSAN Orchestrator and ran locally via the supervisors located on the Raspberry Pi devices. This module allows the device to calculate a local trust score for peer devices based on available trust related data and observed responsiveness of the peer device. In the lab setup, trust related data used are believability scores and anomality scores related to the mining and tunnelling use case and the transportation use case.

A device executing the trust scoring WASM-module first initiates pings to its known peer devices and calculates an initial trust score based on the results of the pings (min/max/average/standard deviation/received packages). Another score is derived from the trust data (believability/anomaly scores). Lastly, the two scores are joined together with weight leaning more towards the final believability/anomaly score, as these scores are related to the core functions of the use cases while the trust score generated from the ping results is related more to the general responsiveness of the devices. These local trust scores are then stored in a local JSON file on the executing device and are fetched by the management server via HTTP requests and used to calculate global trust scores. If the local trust score of a device is too low, it will get blacklisted by the CISSAN management server. The local trust scores are used to calculate a global trust score for the network the devices reside in.

3.6.5 Security and Automated Disaster Recovery Orchestration

The system functionality handling orchestration of security functions in the network and the associated automated disaster recovery behaviour is composed of several distinct sub-functions involving different components active at different operating phases of the CISSAN system.

The different components include CISSAN system components such as the CISSAN Orchestrator and the CISSAN management server as well as external system components such as the optimization solver and the Wireguard VPN component.

Optimal task allocation and deployment

The CISSAN platform distributes security tasks over the network with consideration taken to aspects such as, the security risk of executing a task on a device, task and device capacity demand and supply, network topology, disaster recovery requirements and more.

The operational phase in which the optimized task deployment is identified involves cooperation between the CISSAN Orchestrator, the CISSAN Management Server and the Optimization Solver.

Figure 5 depicts the distribution of the components and the types of interfaces that are used for integration on different protocol layers in the CISSAN platform.

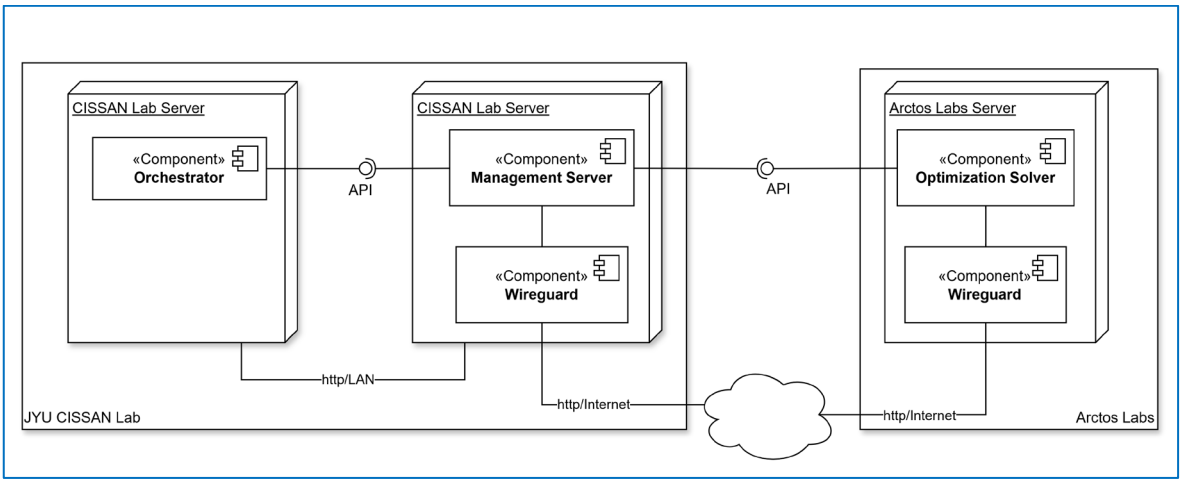

Figure 5. Deployment of components in the CISSAN platform

Details of the optimal task allocation and information exchange performed therein are not provided as they are part of a pending patent application and research paper.

Orchestration of tasks on devices during normal operation

The Arctos Labs Optimization Solver provides the CISSAN Management Server with a template for the optimal task distribution, which the CISSAN Management Server then forwards to the CISSAN Orchestrator. The CISSAN Orchestration Server provisions the tasks to the relevant devices. Task provisioning is described in D5.5, Section 4.2. After the devices have fetched the required software packages, the CISSAN Management Server can request the execution of the tasks. Task execution is described in D5.5, Section 4.3.

Disaster recovery in case of system failures

The CISSAN Orchestration Server continuously monitors the health of the connected devices. In the event of a device failure, the CISSAN Orchestrator relies on the failover devices provided by the Arctos Labs Optimization Solver for the deployment. The CISSAN Orchestration Server's goal is to replace the failed device with a functioning failover device and to switch back to the original device once it is fully operational again. Disaster recovery is described in more detail in the CISSAN deliverable D5.5 report, Section 6.

3.6.6 Threat Intelligence Sharing

Threat intelligence sharing in the lab was implemented primarily as structured security telemetry exchange across the OT network and toward central monitoring, enabling local detections to be externalized in consistent formats. Device observations and trust-related updates were represented as JSON and JSONL to support interoperability between components and straightforward ingestion by central systems. In addition, the project prototypes standards-based CTI export so that selected high-value security events can be represented in an industry-recognized format for broader sharing and tooling compatibility.

The anomaly reports generated by the local monitoring modules, are shared in the following format:

JSON payload
{

“timestamp”: string/number

“anomalies”: [array of objects]

“source_rtu”: string

}

The aggregation is performed at the CISSAN Management Server, where the reports are parsed into JSONL.

Standards-based sharing and trust-centric CTI information follows the STIX 2.1 documentation. These concepts are covered in D7.2, and the related STIX implementation documentation in D5.4. Security and trust management/monitoring data flows are detailed in the Rotor research report.

3.6.7 Automated Incidence Response

The CISSAN Orchestrator maintains a comprehensive device database, tracking the real-time status of each node as Active, Inactive, or Blacklisted. Through periodic health checks, the CISSAN Orchestrator monitors device integrity and adjusts these statuses dynamically. Security incidents, such as device outages or blacklisting reported by the management server, trigger immediate state transitions. When a device is inactive or flagged as blacklisted, the CISSAN Orchestrator updates its

status and initiates an automated redistribution of tasks to designated failover nodes to ensure service continuity. Blacklisted devices are excluded from any future deployments.

3.7 CISSAN Software Framework

The CISSAN software framework has been developed in collaboration with the Liquid AI project. The Liquid AI project investigated how application functionality could be flexibly deployed and managed across different system components using the principles of isomorphic and liquid software [1]. The core idea is that the same application logic should be able to be executed on different devices without the need to rewrite the code for each platform. To support this, WebAssembly was adopted as a common execution format, enabling software modules to be developed independently of the underlying hardware and deployed uniformly across the system [2].

Building on these principles, the orchestrator was implemented as a coordination component, managing deployment and execution of WASMs across a heterogeneous fleet of IoT devices. An actor, represented by the management server, interacts with the orchestrator to request functionalities, which it fulfils by coordinating deployments, execution, and device monitoring. The management server determines an optimized task allocation across devices using an external optimization component and forwards the resulting deployment solution to the orchestrator for execution.

To manage the distributed system, the orchestrator maintains several internal data stores. The Device Database records device metadata and current state information. The Deployment Registry tracks all executed deployments, including participating devices and assigned WASMs. The Package Manager catalogues available WASMs and resolves dependencies to ensure each module can run successfully on its target devices [2, 3]. These stores collectively support deployment planning, execution scheduling, and system monitoring.

Supervisor software runs on each device, providing a local WebAssembly runtime and interfaces for device monitoring and management. The orchestrator communicates with supervisors to deploy workloads, gather metrics, and maintain an up-to-date view of system resources.

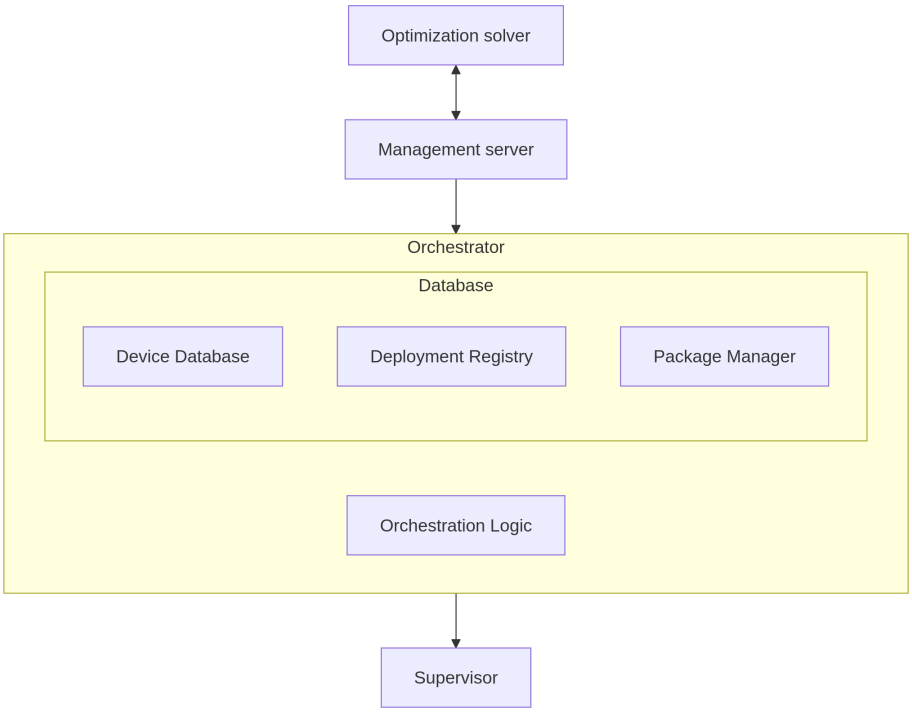

Figure 6. Orchestrator architecture

Based on this architecture (see Figure 6), the Liquid AI orchestrator was extended by the Liquid AI project team as a result of the collaboration between the Liquid AI and CISSAN projects to support automated disaster recovery and distributed security task execution, which is referred to as the

CISSAN orchestrator. The first addition was continuous device health monitoring, which allows the orchestrator to maintain an up-to-date view of device availability.

The deployment template was extended to include pre-defined failover devices. These alternatives are provided by the optimization solver and represent devices capable of executing the same WebAssembly functionality. When a device is marked as inactive based on health monitoring results, the CISSAN Orchestrator transitions the affected deployments into failover handling, updating the deployment by replacing the inactive device with an available failover candidate.

Support for recovery was added in a similar manner. When a previously inactive device is observed to be healthy again, it is reintroduced into deployments where it originally participated, allowing the system to return to its intended execution topology automatically.

In addition, dedicated WASMs were developed to implement security-related functionality, enabling these tasks to be distributed across multiple devices and coordinated by the CISSAN Orchestrator to support CI.

The CISSAN Management Server was introduced to act as the external actor that initiates the CISSAN Orchestrator operations. The motivation for this separation was to clearly distinguish between system-wide decision-making and the execution-oriented responsibilities of the CISSAN Orchestrator. Operations such as defining which tasks should be executed, in what order, and when a deployment should be started must be triggered by an actor with a global view of the system, rather than being handled internally by the CISSAN Orchestrator itself.

In the implemented architecture, the CISSAN Management Server fulfils this actor role. It is responsible for initiating deployments, triggering execution, and interacting with external components such as optimization logic. The management server is also responsible for deciding when a device should be blacklisted. The CISSAN management server is designed to represent a holistic view of the system, making it suitable for use by a system or security administrator who needs visibility into overall system state, device status, and ongoing operations.

The CISSAN Orchestrator, in contrast, focuses on enforcing the requested actions and managing their execution. Its responsibilities are limited to deployment management, execution coordination, device monitoring, and disaster recovery handling. By decoupling these roles, the system maintains a clear separation of concerns: the management server decides what should happen and when, while the CISSAN Orchestrator determines how those actions are carried out reliably in the distributed environment.

3.8 Collective Intelligence (CI)

3.8.1 CI in CISSAN Industrial IoT network

CI is a core architectural theme in the CISSAN industrial IoT (IIoT) network. Rather than treating OT assets as isolated data sources, the platform integrates mechanisms that allow distributed components to share and aggregate security-relevant signals, validate observations, and support coordinated response decisions at runtime. Within the OT network, CI-work is demonstrated through three key solutions, Rotor framework, NodeEye**,** and PASAD.

Rotor’s CI architecture follows a distributed-first model with optional central aggregation, designed explicitly for resource-constrained industrial devices. Each monitored unit acts as an autonomous security participant that produces local observations through embedded anomaly detection. These observations are then exchanged among peers to enable cooperative reasoning, essentially upgrading the local anomaly detection to a collective threat hunt. Devices are not treated as passive sensors, but as collaborating nodes that can cross-check abnormal behaviour and reinforce network-wide situational awareness. At the same time, Rotor incorporates a centralized aggregation layer that collects peer-derived trust updates and anomaly context to maintain a consolidated trust view and enable coordinated response decisions, including trust-based isolation when a device is assessed as unreliable. This architecture enables CI without requiring continuous centralized analytics, while still providing an operator-facing oversight point and integration path to SIEM/CTI tooling. A detailed description of the framework’s components, workflows, and experimental

validation is provided in the CISSAN deliverable D5.4 report and in the Rotor framework research report. An overview of the framework is displayed in Figure 7.

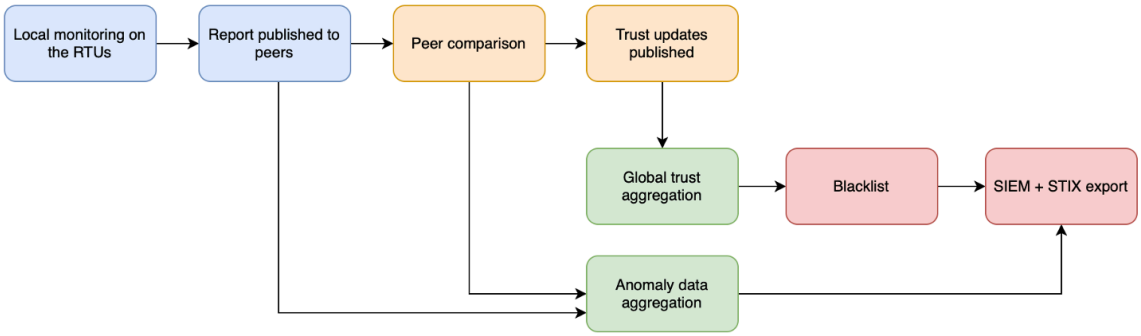

Figure 7. Overview of the Rotor framework

3.8.2 CI in CISSAN IoT network

IoT devices are represented by Raspberry PIs in the CISSAN network. Collective Intelligence is demonstrated with the use of Liquid AI software on these Raspberry PI devices. The Liquid AI solution consists of an orchestrator directing multiple supervisors located on IoT devices running WASM-modules which are distributed via the CISSAN Orchestrator. The modules are capable of sending the results of their execution forward to other modules, enabling splitting a larger execution to smaller components for resource constrained devices. In the case of CISSAN, separate WASM-modules for calculating anomaly, believability, and trust scores have been developed. These modules are distributed to the CISSAN lab IoT network portion and executed in succession with each other. This way, the Raspberry PI devices will be able to split the task of calculating device trust scores for their peer devices which will be shared with the management server of the platform and used to calculate global trust scores for the network. In case an individual device's trust score drops below an acceptable threshold, the device is blacklisted. The blacklisted device's tasks are then redistributed to other devices on the IoT network. This redistribution allows the network to continue generating security data even if some devices are lost.

NodeEye is an embedded real-time fault detection and classification system designed to run directly on an RTU-class microcontroller based on the Techinova's hardware platform. It acts as an intelligent edge node for 3-phase feeder monitoring, continuously analysing operational electrical signals and reporting abnormal behaviour with minimal latency and without reliance on cloud processing.

Architecturally, NodeEye implements a two-stage AI pipeline optimized for constrained hardware environments. The first stage is a lightweight voltage anomaly detector that continuously evaluates short sliding windows of three-phase voltage data. Rather than relying on fixed relay-style thresholds, it models normal feeder behaviour and computes a deviation score for each time window. When the deviation remains within learned bounds, the system publishes status and health metrics only. When the score exceeds a defined threshold, the window is marked as anomalous and escalated for further analysis.

The second stage is a compact fault classifier that activates only when an anomaly is detected. This model analyses the corresponding three-phase current window and assigns a fault category such as line-to-ground, line-to-line, double-line-to-ground, three-phase, or three-phase-to-ground together with a confidence value.

Operationally, NodeEye works as best if used on multiple secondary substations collectively. Using it in this method, the data can be collected and used to pinpoint the exact substation that is faulty. Because multiple substations can have the same or very similar voltage fault, which NodeEye will flag, but multiple substations will not have the same current fault. So collectively they are showing where the fault is and who is at fault.

From a systems perspective, the key advancement lies in achieving reliable AI inference on constrained RTU hardware. The neural networks are deliberately compact, with small memory footprints and predictable execution timing, allowing them to run within the flash, RAM, and throughput limitations of a microcontroller platform. This enables deterministic real-time behaviour, stable message handling, and continuous monitoring at operational sampling rates, all without the need for industrial PCs or remote compute resources.

3.9 Device, Edge, and Agent Layers

The IoT devices of the CISSAN platform consists of five Raspberry PI devices connected to a Teltonika RUTx10 router which is connected to the L2 Switch. There are two different types of Raspberry PI devices, three 4B1 models and two 2B1 models all running Debian GNU/Linux 12 (bookworm) distros. The Raspberry PI devices are used to represent IoT devices in use cases one (1), three (3) and five (5). This is handled by executing WASMs with the Liquid AI software solution consisting of an CISSAN Orchestrator running on the Orchestration server on the platform and supervisors running on the Raspberry PIs. The supervisors themselves are running in tmux windows on the Raspberry PIs.

In the CISSAN platform, device security is ensured as follows:

Network-Level Controls

- VLAN separation of OT and IT.
- Firewall segmentation (5G router).
- Restricted communication paths (no RTU ↔ Pi communication).

Access Control

- SSH-based administrative access.
- FreeRADIUS-based authentication where supported.
- Removal of default credentials.
- Protected management interfaces.

System Hardening

- Minimal OS installations on possible systems.
- Local firewall enforcement (default deny).
- Vendor firmware on RTUs.
- Secure Boot on virtualization layer (Hyper-V + Gen2 VMs).

Monitoring

- Centralized logging and SIEM (Wazuh).
- Alerting enabled.

The following security features were not implemented in the project, and are planned to be addressed post-project:

- Hardware root of trust (TPM) on edge devices.
- Remote attestation.
- Mutual TLS device identity.
- Full transport encryption for all OT protocols.
- Automatic patching on edge devices.
- Dedicated network IDS/IPS.

Agents

The proof-of-concept (PoC) implementation of the fully distributed, agent-based anomaly detection system is entirely software-based (for more information, see Delivery 5.4, section 2.2.5). It is running on any linux-based systems. The demonstration run was executed on a Linux (Debian 12) platform powered by an AMD Ryzen 5 processor and operated using synthetically generated data.

3.10 CISSAN Interfaces and Communication Mechanisms

The interfaces and communication mechanisms used in the CISSAN platform, and its use case solutions are summarized below.

3.10.1 Internal Service Interfaces

The Mattersoft GPS anomaly detection system interfaces are summarized in Table 1.

Table 1. Mattersoft GPS anomaly detection system interfaces

Interface (Producer → Consumer)	Purpose	Mechanism / Protocol	Data Objects	Trigger / Frequency	Security Controls	Implementation Notes
Mattersoft GNSS Data Producer → MQTT Broker	Publish live vehicle telemetry for research consumption	MQTT publish	ITxPT GNSSLocation, ITxPT AVMS Runmonitoring	GNSSLocation: ~1 Hz per active vehicle; Runmonitoring: event-driven	TLS; authenticated publisher identities; broker-side validation	Operational producers publish into topic namespace itxpt/{operator}/{vehicle}/{message_type}
MQTT Broker → CISSAN Ingestion / Connector	Provide read-only fleet data stream into CISSAN	MQTT subscribe (QoS 1)	GNSSLocation stream; Runmonitoring stream	Continuous; at-least-once delivery	TLS; per-client credentials; per-topic ACLs; explicit deny of publish/retain/admin	CISSAN is restricted to subscription on GNSSLocation + Runmonitoring topics only; no vehicle control supported
CISSAN Ingestion / Parser → CISSAN Storage	Persist and index received transport telemetry	Internal service call (implementation-dependent)	Normalized GNSSLocation + Runmonitoring events; metadata	Streaming ingestion	Access control; integrity controls; retention policies	Storage enables later analysis and replay for anomaly detection research

CISSAN Analytics → Trust / Scoring	Provide anomaly risk as an input to trust scoring	Internal API	Anomaly risk score;	Event-driven or batch window	Service-to-service auth; least privilege	UC1 supports research analysis

3.10.2 IoT and OT Device Interfaces

The GPS anomaly detection system IoT device interfaces of Use Case 1 are summarized in Table 2.

Table 2. Use Case 1 GPS anomaly detection system IoT device interfaces

Capability / Interface Contract	**Profile A: Lightweight IoT Telemetry**	**Profile B: OT via Gateway / Collector**	**Profile C: Managed Node / Edge Agent**	**Profile D: Legacy / Read-Only**
Telemetry / metrics upload	No	No	No	Yes (via MQTT subscribe)
Security event reporting	No	No	No	Limited
Health / heartbeat	No	No	No	Best effort
Local anomaly detection	No	No	No	No / limited
Evidence attachment	No	No	No	No
Trust score input	No	No	No	Yes (telemetry)
Trust score receipt	No	No	No	No
Config updates	No	No	No	No
Remote commands / actuation	No	No	No	No
Quarantine / blacklisting enforcement	No	No	No	No enforcement path to vehicles in this implementation
Secure channel (TLS/mTLS)	N/A	N/A	N/A	Yes (TLS)
Payload signing (PQC)	No	No	No	No

Payload encryption (PQC)	No	No	No	No
Offline / store-and-forward	No	No	No	No
Supported integration mode	N/A	N/A	N/A	Passive / read-only ingestion

UC1 does **not** include any CISSAN-controlled actuation path to vehicles; the integration is strictly for secure telemetry ingestion and research analysis.

3.10.3 External Systems Integration

The interface related to the integration of the CISSAN platform with the Arctos Labs Optimization Solver is summarised in Table 3.

Table 3. Interface related to the integration with the Arctos Labs Optimization Solver

External System / Interface	Purpose	Interface Type	Data Exchanged	Security & Trust	Configuration / Extensibility
Optimization Solver	Provide CISSAN Orchestrator with optimized task to device allocation	REST API	Task and network descriptors, task to device maps	Cryptography	-

Detailed description of this interface can be found in the CISSAN deliverable D6.2 - Annex 3.

Councilbox Blockchain System

The Councilbox Blockchain System (Eventchain) is a purpose-built distributed blockchain for IoT device governance within the CISSAN platform. The CISSAN management server interacts with Eventchain through a stateless REST API (the CISSAN Metadata API) to record device registrations, trust score updates, anomaly detection events, and blacklisting decisions. Every transaction is signed with ECDSA secp256k1 and validated through 51% Byzantine Fault Tolerant (BFT) consensus before becoming part of the permanent ledger.

The interface related to the integration of the CISSAN platform with the Councilbox Eventchain System is summarised in Table 4.

Table 4. Integration with the Councilbox Eventchain System

External System / Interface	Purpose	Interface Type	Data Exchanged	Security & Trust	Configuration / Extensibility
Metadata API	Device lifecycle management: registration,	REST API over HTTPS	Device registration records, trust score updates	JWT (HS256) bearer token authentication; all BKVS	Configurable JWT expiration and rate-limiting

	trust score updates, anomaly detection event recording, and device blacklisting		(score, confidence, contributing factors), anomaly detection events (type, severity, method), blacklist events (reason, threshold, triggering anomalies)	writes signed with ECDSA secp256k1; bcrypt password hashing; rate limiting on sensitive endpoints	parameters. BKVS namespace derived from the API signing key.
Blockchain Explorer	Read-only blockchain visualisation for audit and inspection of recorded events, blocks, and cryptographic proofs	HTTPS (web UI)	Blocks, transactions, Merkle tree structures, consensus validation status, cryptographic proofs, anchoring status	Read-only access; no write capability exposed; served over HTTPS	Explorer instance configurable per node. Supports inspection of any block height or transaction hash

The complete API specification is provided in D6.2. The full system architecture is described in the CISSAN deliverable D4.4 report.

Clavister Smart Grid Monitoring System

The interface related to the integration of the CISSAN platform with the Clavister Smart Grid Monitoring System for local anomaly detection for relay fault detection and correlation is summarised in Table 5.

Table 5. Integration with the Clavister Smart Grid Monitoring System

External System / Interface	Purpose	Interface Type	Data Exchanged	Security & Trust	Configuration / Extensibility
Relay Fault Correlation System	Export fault events with clear indication of fault location in the smart grid.	MQTT publish-subscribe API	Anomaly, sensor(s) affected, affected substation(s), fault origin	TLS-protected MQTT where available	Configurable MQTT broker IP/Port

Detailed description of this interface can be found in the CISSAN deliverable D6.2 - Annex 4.

Mattersoft GPS anomaly detection system

The interfaces related to the integration of the CISSAN platform with the Mattersoft GPS anomaly detection system is summarised in Table 6.

Table 6. Integration with the Mattersoft GPS anomaly detection system

External System / Interface	**Purpose**	**Interface Type**	**Data Exchanged**	**Security & Trust**	**Configuration / Extensibility**
Mattersoft GNSS Data Source (Public Transport Fleet)	Provide near real-time GNSS positioning and operational context for UC1 anomaly detection and demo use case workflows	**MQTT publish–subscribe API** (ITxPT-compliant message payloads)	ITxPT **GNSSLocation** (~1 Hz) and ITxPT **AVMS Runmonitoring**(event-driven trip context)	TLS-protected MQTT where available; authenticated client access; topic-level read-only permissions for CISSAN consumers; integrity is the primary objective (data is non-sensitive)	Topic namespace supports scalable fleet expansion; subscription filtering by vehicle and message type; supports parallel consumers (e.g., visualization + anomaly detection)
University of Jyväskylä Visualization / Mapping UI (OpenStreet Map-based)	Display live vehicle positions and highlight anomaly-affected vehicles during demo choreography	MQTT subscription + internal UI API	Vehicle position stream, derived anomaly status (e.g., “normal” vs “suspected spoofing/jamming”)	Access controlled within lab environment; demo-grade controls; focus on preventing injection of false anomaly flags into UI	UI can be extended to show per-vehicle details (trip, confidence score, anomaly category) and historical playback
CISSAN UC1 Anomaly Detection Component	Analyze incoming GNSS streams and detect spoofing/jamming anomalies for the transport use case	Internal processing pipeline; input via MQTT; output via MQTT/REST	Raw GNSSLocation, Runmonitoring context, derived anomaly events, confidence values	Component access restricted; anomaly outputs are treated as integrity-critical; events logged for traceability	Algorithm parameters configurable (thresholds, correlation windows); supports iterative improvements without breaking the MQTT contract
GTFS Static Dataset Provider (Transit Schedule Data)	Provide route geometry, stop sequence, and timetable reference for route-aware validation	File-based import (GTFS) / internal dataset interface	GTFS static files (routes, trips, stops, shapes, stop_times)	Non-sensitive data; integrity ensured via controlled dataset managemen	Supports multiple operators and GTFS versions; enables reproducible research runs

	and anomaly correlation			t and versioning	
Workshop Attack Simulation / Test Harness	Trigger controlled spoofing/jamming scenario for the UC1 demo choreography	Local simulation tool; controlled injection into test environment	Attack trigger events; simulated abnormal GNSS behavior patterns	Restricted to lab; not exposed to operational systems; separated from production GNSS feed	Scenario library can be extended (gradual drift, hard jump, total loss, intermittent jamming)
CISSAN Platform / Management Services (JYU Lab)	Provide orchestration, storage, and correlation services for demo workflows and future scaling	Internal service interfaces (REST / MQTT / storage)	Stored GNSS telemetry, anomaly logs, demo outputs	Demo-level access control; logging enabled for traceability; target security baseline defined in UC1 SRS	Supports future integration to SIEM/STIX exports and extended trust scoring mechanisms
External SIEM / SOC (optional / future)	Export anomaly events for security monitoring and incident workflows (beyond demo)	REST API / message bus	Alerts, incidents, metadata	mTLS / role-based access (target capability); signed payloads (optional)	mTLS / role-based access (target capability); signed payloads (optional)

3.10.4 Data Exchange Formats and Protocols

The data exchange formats and protocols used in CISSAN are summarized below.

HTTP: The CISSAN orchestration server and IoT devices communicate with HTTP. Deployment manifests, execution status, and device metadata are exchanged in JSON format. Device discovery is handled via mDNS, allowing the CISSAN Orchestrator to identify available nodes automatically.

JSON: JSON is a lightweight open standard file format and data interchange format consisting of name-value pairs and arrays that uses human readable text. It is used to store local trust scores generated by the trust scoring WASM-module.

MQTT: The MQTT protocol is used to establish a connection with the Mattersoft GPS system to obtain the GNSS data for the transportation use case, as well as to use it as a messaging protocol for communications between various components of the CISSAN platform and external systems that need to exchange telemetry data. In particular, an MQTT broker is used to ingest RTU trust and telemetry and to publish trust scores, deployment snapshots, and device blacklist events. The CISSAN Management Server GUI subscribes to MQTT for live state.

STIX: The STIX threat report format is a structured representation of cyber threat intelligence at the CISSAN Management Server, which is based on the STIX 2.1 standard and is used within the CISSAN platform to consolidate, represent, and share the gathered information in an interoperable manner.

PQC: While PQC has not yet been adopted in the current CISSAN platform implementation, it is intended to be integrated post-project to enable secure communications between the CISSAN platform and external system components, including partner systems, particularly for the encryption of payload data and digital signing of exchanged messages.

4 Component Implementation Definition

4.1 CISSAN Management Server Implementation

The CISSAN Management Server runs as a single daemon process (systemd or Docker) on the lab host. It keeps durable state in data/state/ (device registry, port isolation list, task ledger, local trusts) and uses data/cache/ for transient outputs. It connects to the CISSAN Orchestrator and optimization solver for deployment, to device supervisors over HTTP for task execution, and to the MQTT broker to ingest RTU trust and telemetry and to publish scores, deployment snapshots, and blacklist events. The management GUI subscribes to MQTT for live state; the daemon does not serve the GUI. Trust-based automated protection is handled by calling the port blacklister when a device's trust score falls below a configured threshold.

4.2 CISSAN Orchestration Server Implementation

The CISSAN Orchestration Server was deployed in the CISSAN lab as a Docker container on an Ubuntu Server. Persistent data is stored in a MongoDB instance, with separate collections for devices, deployments, and modules, corresponding to the device database, deployment registry, and package manager. The CISSAN Orchestrator exposes a REST-based API through which the Management Server can submit deployment requests and trigger application execution.

4.3 SCADA Server Implementation

The SCADA runs a COPA-DATA Zenon Service Engine and Engineering to both build and run the SCADA server. It runs in the same VLAN as the RTU's to poll them for mockup data.

4.4 SIEM Implementation

A simple SIEM deployment with Wazuh is implemented as an all-in-one architecture combining analysis, storage, and visualization components.

The Wazuh server collects telemetry from agents distributed on all IoT devices, applies decoding and correlation rules, and generates security alerts. Alerts and events are forwarded to the Wazuh indexer, which provides indexed storage and fast querying. The Wazuh dashboard connects to the indexer to deliver real-time monitoring, alert triage, and investigation workflows.

The Wazuh server also collects alerts from the Rotor software via a proxy on the CISSAN Management Server. The alerts are collected there via MQTT, then sent to Wazuh via its agent, whereas a custom decoder and custom ruleset are used to interpret the alerts.

Conclusion

This report defines the implementation of the CISSAN architecture by describing the elements of the distributed system, their roles, and the interfaces connecting the various components of the system into a coherent and interoperable platform. This demonstrates the implementation of the CISSAN framework through the CISSAN platform and the associated software framework. It represents the updated CISSAN architecture, building on the initial architecture definition document provided in the deliverable D2.1 report and the lessons learned throughout the CISSAN project. The architecture was updated to include the new requirements and capabilities, including the use of WebAssembly-based isomorphic software components to support execution in heterogeneous environments and the inclusion of the security and disaster recovery orchestration components contributed by the Liquid AI project team as part of the collaboration between the CISSAN and Liquid AI projects. The software architecture was also a result of the collaboration and evolved from the initial architecture to the more modular, flexible, and operationally robust architecture as a result of the collaboration.

The implementation of the CISSAN architecture also incorporates the security-by-design and zero trust concepts to ensure that the system's security considerations are integrated into the CISSAN architecture as opposed to being an afterthought. The use of modular components with well-defined interfaces and a common software framework also helps to support scalability, maintainability, and adoption, which are critical in ensuring a gradual transition to a more distributed and collaborative architecture as opposed to the traditional centrally managed legacy architecture.

The implementation definition of the CISSAN architecture presented in this report provides a robust and extensible technical foundation for the CISSAN platform and the associated CISSAN framework. It also provides a foundation for the CISSAN project’s objectives, including the achievement of greater decentralisation, interoperability, and the support for integration of future technologies such as post-quantum cryptography. It provides a practical and sustainable foundation for the development of collective intelligence-based cybersecurity solutions for the real world.

References

[1] University of Jyväskylä, “Liquid AI for 6G software,” [Online]. Available: https://www.jyu.fi/en/projects/liquid-ai-for-6g-software. [Accessed 24 2 2026].

[2] P. Kotilainen, V. Heikkilä, K. Systä, T. Mikkonen, M. Younas, I. Awan and T.-M. Grønli, “Towards Liquid AI In IoT With WebAssembly: A Prototype Implementation,” in *Mobile Web and Intelligent Information Systems*, Marrakech, 2023.

[3] P. Kotilainen, V. Järvinen, T. Autto, L. Rathnayaka and T. Mikkonen, “Demonstrating Liquid Software in IoT Using WebAssembly,” in *24th International Conference on Web Engineering*, Tampere, 2024.

Annexes

The confidential annexes of this document are listed below.

A1: D2.3 - Annex 1: UC1 Security Requirements Specification

A2: D2.3 - Annex 2: UC2 Security Requirements Specification

A3: D2.3 - Annex 3: UC3 Security Requirements Specification

A4: D2.3 - Annex 4: UC4 Security Requirements Specification

A5: D2.3 - Annex 5: UC5 Security Requirements Specification