Distributed Intelligence for Enhancing Security and Privacy of Decentralised and Distributed Systems (Di4SPDS)

Topic: Chist-era 2022 — Security and Privacy in Decentralised and Distributed Systems (SPiDDS)

Deliverable D.2.2: Di4SPDS Reference Architecture and Technical specification

Work Package	Requirements Analysis & Framework Reference Architecture
Delivery Date	30/06/24
Responsible Partner	LUT
Authors	Prabhat Kumar (LUT) A. Santos-Olmo (UCLM) M. Tuncay Gençoğlu(FU) Kandaraj Piamrat (NU)
Contributors	Prabhat Kumar (LUT) Md Raihan Uddin (LUT) Gauri Shankar (LUT) A. Santos-Olmo (UCLM) M. Tuncay Gençoğlu(FU) Özgür Karaduman(FU) Haluk Eren (FU) Kandaraj Piamrat (NU) Houssem Jmal (NU)
Version	0.3
Reviewer Name	Kandaraj Piamrat (NU)

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.0	07.03.2024	Table of Contents	Shareeful Islam (LUT)
0.1	16.05.2024	3.1 Component 1- Blockchain-based cross-domain access control Specification	M. Tuncay Gençoğlu, Özgür Karaduman, Haluk Eren (F.U.)
0.1	16.05.2024	3.3 Dynamic Risk Management	A. Santos-Olmo (UCLM)
0.1	18.05.2024	3.2 Component 2- Multi-agent and Self-aware Collaborative Intrusion Detection Systems	K. Piamrat, H. Jmal (NU)
0.1	21.05.2024	1. Introduction 2. Di4SPDS - Conceptual view 4. Di4SPDS Reference Architecture	Prabhat Kumar, Md Raihan Uddin, Gauri Shankar (LUT)

List of Abbreviations and Acronyms

Abbreviation / Acronym	Meaning
DDS	Decentralized and Distributed Systems
ICT	Information and Communication Technology
CA	Certificate Authority
P-to-P/P2P	Peer-to-Peer
CIDS	Collaborative Intrusion Detection System
PKI	Public Key Infrastructure
ACM	Access Control Model
MA	Mutual Authentication
IPFS	Inter Planetary File System
MFA	Multi-Factor Authentication

RBAC	Role-Based Access Control
SSL	Secure Socket Layer
TLS	Transport Layer Security
HIPPA	Health Insurance Portability and Accountability Act
GDPR	General Data Protection Regulation
ECC	Elliptic Curve Cryptography
FL	Federated Learning
IID	Independent and Identically Distributed
CPS	Cyber-Physical Systems
CVE	Common Vulnerabilities and Exposures
CAT	Controls, Assets, and Threats
MOF	Meta-Object Facility
UML	Unified Modeling Language
KRI	Key Risk Indicators
RPG	Risk Pattern Generator
RAMG	Risk Analysis and Management Generator
DRM	Dynamic Risk Management
J2EE	Java 2 Platform, Enterprise Edition
JSON	JavaScript Object Notation
CSV	Comma-Separated Values
PCAP	Packet Capture
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol

Executive Summary

Insert text

Table of Contents

1. Introduction

In today's rapidly evolving digital landscape, Decentralized and Distributed Systems (DDS) have become essential components of the ICT infrastructure, enhancing business capabilities with extensive computing and processing power. While these systems effectively meet critical business needs for bandwidth, data storage, and computing resource utilization, their widespread adoption also increases susceptibility to a variety of security and privacy threats. From vulnerabilities that allow threats to propagate across interdependent resources to sophisticated cyber-attacks like the 2016 Mirai attack, the need for robust cybersecurity measures in DDS is more pressing than ever. The Di4SPDS project is designed to confront these escalating challenges head-on by implementing a state-of-the-art security framework tailored for DDS. The project introduces innovative solutions that not only address current vulnerabilities but also anticipate and mitigate evolving cyber threats. This is achieved through a synergistic approach involving blockchain technology, smart contracts, multi-agent systems, and federated learning, all orchestrated to enhance the security, privacy, and operational efficiency of decentralized environments.

1.1 Purpose of the Project:

The Di4SPDS initiative aims to transform how security is implemented in decentralized systems by developing:

- A blockchain and smart contracts-enabled cross-domain authentication and access control scheme.
- A multi-agent and self-aware collaborative intrusion detection system.
- A dynamic and sustainable risk management practice.

These components are designed to ensure automation, immutability, transparency, and trust while providing robust resistance to a variety of known and emerging cyber threats.

1.2 Goals of the Deliverable:

This deliverable provides comprehensive documentation of the Di4SPDS project, detailing the technical specifications of each component, conceptual views, and the reference architecture. It serves to articulate the methodologies, technologies, and strategies developed to fortify DDS against cyber threats while also fostering sustainability in cybersecurity practices.

1.3 Expected Impact:

The Di4SPDS project is anticipated to significantly elevate the security posture of DDS by enabling more effective prevention, detection, and response mechanisms to cyber incidents. By integrating technologies such as blockchain and federated learning, the project not only improves security but also ensures that these enhancements are sustainable and efficient in resource consumption. Ultimately, Di4SPDS will serve as a benchmark for future cybersecurity initiatives in decentralised environments.

2. Di4SPDS - Conceptual view

2.1 Overview

The development of the Di4SPDS framework aims to enhance the security and privacy of the decentralized and distributed system and provide a secure communication channel between participating domains. The framework adopted cross-domain communication for authentication and access control using smart contracts on the blockchain network. The framework is integrated with the blockchain-enabled, multi-agent, self-aware, collaborative federated learning model to provide an intrusion detection mechanism. Furthermore, dynamic risk management will add to this framework to identify the threats and risks associated with achieving sustainable security and provide countermeasure strategies to defend against those threats and risks. Dynamic risk management will provide advanced vulnerability exploitability prediction using machine learning. The development of this framework aims to provide enhanced, sustainable, secure, decentralized and distributed system solutions for healthcare, businesses, industries, and society, ensuring comprehensive protection of data, privacy, and communications across all sectors.

2.2 Key components

There are three main components in Di4SPDS:

2.2.1 Blockchain-based cross-domain access control

This component is responsible for performing mutual authentication between the participating entities. The design of this component involves blockchain, smart contract, authentication and access control protocols and user entities. The private blockchain will deploy at the local organization/domain with a smart contract that will provide authentication and access control on users' requests locally. Each private blockchain has its own private edge servers for storing the identity of the user and local data using an interplanetary file system along with a log of the

communications over the network. These private blockchains form a network from the different domains using a consortium blockchain for cross-domain sharing data and communication. Each participating private blockchain must be registered and accept the mutual authentication agreement in this consortium blockchain. This consortium blockchain manages the cross-domain authentication and access control through a smart contract and has a network of peer-to-peer connected cloud servers with an interplanetary file system that stores the cross-domain communications and identities of the cross-domain entities from the consortium network.

The process of this component starts with the entity (user/device) being registered. The registration is performed at a local private blockchain through a certificate authority (CA). this CA generates a digital certificate (public key) and an authentication code (secret key) using a public key cryptography protocol. This digital certificate will identify the ID of the entity associated with a role for communication or data access on the network. After successful registration, the user needs to log in to the network using both keys to prove the identity, and this process is handled by the local smart contract by verifying the user's identity. The successful login of the user provides an interface to communicate with other entities and a channel to access the data from the edge server. Whenever the user requests data from the edge server, its identity and associated role will first be verified by the smart contract. Then, based on the request and specified role, a smart contract automatically detects the level of access to the particular and takes further action. If the user request and level of access are matched, then the smart contract provides an authentication interface to access the data, where the user has to provide a digital certificate. After successful authentication, the smart contract authorized the user to access the data from the server. Further, when a user requesting for cross-domain communication or data access, the process is handled through the consortium blockchain. In cross-domain communication, the first consortium blockchain verifies the registration of both private blockchains (requesting and target). Upon successful verification of the private blockchains, the registration of the user is verified and validated by requesting local blockchain through the smart contract of the consortium blockchain. After verifying and validating the user consortium, the smart contract provides authorization based on the ID and associated role to access the data from the edge server from another domain or the peer-to-peer cloud server. Furthermore, all local communication, authentication, and authorization are logged in the edge server of the local private blockchain. Also, for all communication, authentication and authorization are goose through consortium blockchain logged in the peer-to-peer(P-to-P) cloud server. These logs will be fuel for component 2, “Multi-agent and Self-aware Collaborative Intrusion Detection Systems”, at both the edge server and the P-to-P cloud server to identify the intrusion in a Collaborative manner.

2.2.2 Multi-agent and Self-aware Collaborative Intrusion Detection Systems (MSCIDS)

This component will monitor and analyze the network data in real-time to detect intrusion and related threats in Decentralized and Distributed Systems (DDS). It will be based on semi-asynchronous federated learning and blockchain, which will be responsible for selecting clients, assigning the appropriate number of local updates to clients, and aggregating their local models in a timely manner. Furthermore, the optimal value of participating workers will be identified to minimize the training time in accordance with the communication budget, participating workers, edge server heterogeneity, and data distribution. The blockchain ledger will be considered to store the model parameters. Finally, based on the final model, the collaborative intrusion detection system (CIDS) will correlate the data using predefined rules and parameters to understand the possible attacks and support coordinate response based on the multi-agent architecture.

2.2.3 Dynamic Risk Management and Communication and Sharing (DRMCS)

This component will undertake the responsibility to quantify the risks and will propose suitable mitigation strategies to tackle the threats and related risks, aiming to achieve sustainable security. The risk management method considers various data sources to analyze the vulnerability exploitability for the risk assessment activities using machine learning models. The threat and risk information will be shared among different systems while preserving privacy using a federated learning approach.

2.3 Relevant Legal Requirements

Di4SPDS project has to follow several regulatory requirements under the EU AI Act and the EU Cybersecurity Act. These requirements ensure compliance, security, and transparency in the design and development of the system. Below is a summary of the specific legal articles that could apply to the Di4SPDS project.

EU AI Act:

- **Article 6: Classification of High-Risk AI Systems:** In the Di4SPDS project, there are requirements for developing an intrusion detection system and dynamic risk assessments and the experimental scenario will be a distributed and decentralized healthcare system

which is categorized as a high-risk. This requires the project to consider additional obligations in terms of safety, reliability and ethical considerations.

- **Article 9: Risk Management System:** A systematic risk management practice is required to assess, mitigate, and monitor risks throughout the lifecycle of AI components, including federated learning and intrusion detection mechanisms.
- **Article 10: Data and Data Governance:** In the project development, Data usage in the federated learning process and blockchain systems are required to be free from bias and handled securely and transparently to ensure fairness and reliability.
- **Article 13: Transparency and Provision of Information:** The developed AI system must be transparent in its operation, providing stakeholders with clear information about decisions (e.g., intrusion detection or access control authorizations).

EU Cybersecurity Act:

- **Article 51: EU Cybersecurity Certification Framework:** In the Di4SPDS project, there is requirements to develop different modules, for example, blockchain-based access control, federated learning models, and intrusion detection systems. It is recommended that these modules need to be certified under the EU cybersecurity framework to validate their security and resilience.

3. Di4SPDS Individual Component Technical Specification

3.1 Component 1- Blockchain-based cross-domain access control Specification

The secure and efficient sharing of medical information across different healthcare providers is a critical challenge in the digital age. Traditional access control systems, which are often centralized, can suffer from security vulnerabilities, inefficiencies, and lack of interoperability. The Blockchain-Based Cross-Domain Access Control Specification for Medical Information leverages the decentralized, immutable, and transparent nature of blockchain technology to address these challenges. This specification outlines a framework designed to enhance the security, privacy, and interoperability of medical data sharing, ensuring that only authorized entities can access sensitive patient information. By utilizing blockchain, this approach aims to provide a robust and trustworthy solution for managing medical information across diverse healthcare domains.

3.1.1 Overview of the Blockchain-based cross-domain access control Specification

The healthcare sector is inundated with an overwhelming amount of data, encompassing billing information, medical research, and patient history, making systematic and structured

management increasingly challenging. Secure data sharing is essential for healthcare providers and related entities to validate information accuracy, a critical component for ensuring secure medical services. Currently, most health-related information is stored in Electronic Health Records, yet these data remain susceptible to various attacks [FU1]. The complexity of securely sharing health information at the right time adversely impacts patient care, with many patients concerned about the confidentiality and privacy of their health data.

The widespread adoption of Medical Information Systems must be trusted, secure, and efficient to facilitate the seamless exchange of health information among patients, and health organizations. Additionally, issues such as drug piracy and misuse pose significant threats to patient safety, as counterfeit or pirated drugs can be lethal. Ensuring the proper use and management of prescriptions and drugs is thus another critical concern in healthcare.

Blockchain technology, often hailed as revolutionary and disruptive, offers promising solutions for these challenges [FU2]. A blockchain-based healthcare solution ensures patient ownership of their health data and leverages well-defined cloud services to achieve high availability, fault tolerance, privacy, and trust. Blockchain's secure authentication mechanisms prevent unauthorized access to health records, underscoring the necessity of implementing blockchain technology in healthcare. The proposed Blockchain cross domain model is illustrated in Figure 1.

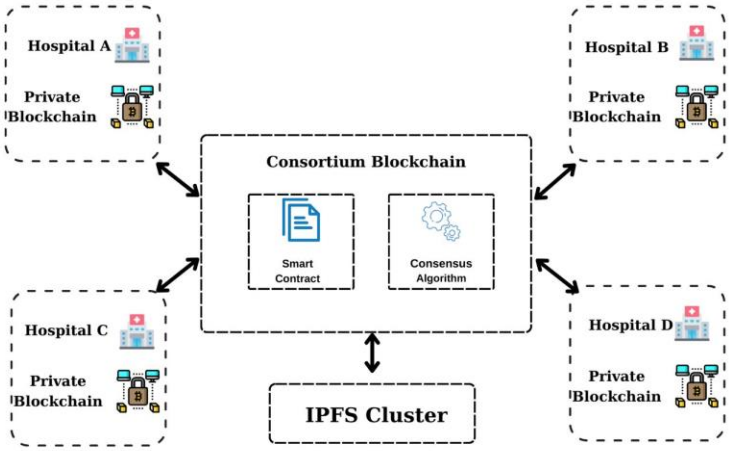

Figure 1**.** Blockchain cross-domain model using private blockchain for sensitive internal data operations within each hospital and managing cross-domain data exchanges between the hospitals via a consortium blockchain.

The main difficulties experienced in conventional systems are:

- **Interoperability of Data:** There are difficulties in sharing and accessing data due to data incompatibilities between healthcare providers.
- **Availability of Basic Data for Doctors:** Incomplete or inaccessible health records make it difficult for doctors to access patients' complete medical histories.
- **Limited Access to Patients' Medical Information:** Patients cannot access their health information fully, making it difficult for them to monitor their treatment process and health status.
- **User Privacy and Data Integrity:** Traditional systems are vulnerable to data fraud, unauthorized access, and compromised data integrity.
- **Medicine Traceability:** Tracking medicines in the supply chain and detecting counterfeit medicines can be difficult.

Additionally, health information stored and accessed through traditional Medical Information Systems can be easily altered for fraudulent purposes [FU3]. Blockchain technology can solve these problems in several ways:

- **Interoperability of Data:** Blockchain increases the interoperability of data as it offers a standardized and distributed ledger system. All healthcare providers have access to data in the same ledger.
- **Data Immutability and Security:** Data on the blockchain can never be modified or deleted, ensuring data integrity and security. Every transaction is verified and encrypted by all nodes in the network.
- **Decentralized Structure:** Since Blockchain has a decentralized structure, it is not subject to the control or ownership of a single party. This makes the system more resilient to faults and attacks.
- **Anonymity and Privacy:** Blockchain encrypts and anonymizes data using Public Key Infrastructure (PKI). Users' identity information is protected by pseudonymity.
- **Patients' Access to Their Data:** Patients can securely access their health information via blockchain and share this information with their healthcare providers.
- **Medicine Traceability:** Blockchain makes it easier to track medicines in the supply chain and detect counterfeit medicines. Because every step is recorded, the origin and distribution process of medicines can be tracked transparently.

These features of blockchain technology can significantly reduce security, integrity, accessibility, and interoperability problems in health information systems. Blockchain cross domain Access Control Model and Mutual Authentication are illustrated in Figure 2.

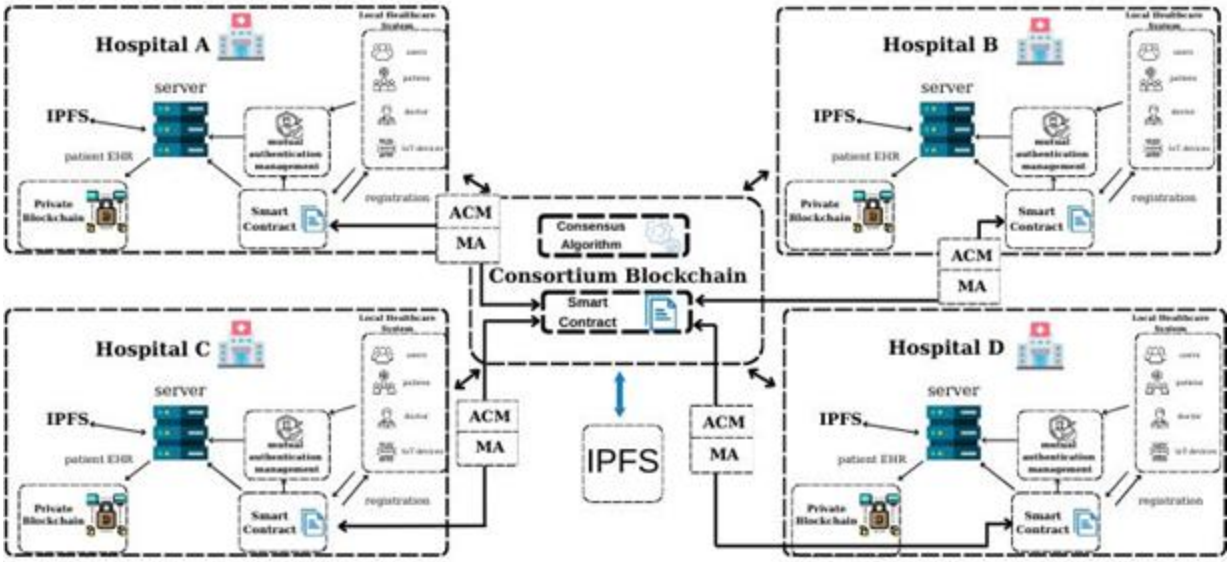

Figure 2**.** Blockchain cross domain Access Control Model (ACM) with Mutual Authentication (MA) including healthcare providers, patients, and administrative staff across the hospitals.

In Figure 2, hospitals represented by A, B, C and D are configured to each have their own private blockchain network. Each hospital ensures security among users within its domain using a mutual authentication system. This mutual authentication system manages the identity verification and authorization of different types of users, such as hospital staff, doctors, nurses, and patients. User authentication processes are carried out through digital certificates issued by a central certificate authority (CA). Each user must have this digital certificate to access the hospital system. For example, when a doctor working at Hospital A registers for the first time, they receive a digital certificate from the CA. This digital certificate verifies the doctor's identity and permissions within the hospital. When the doctor wants to log into the hospital system, they use this digital certificate. After the system validates the certificate and the doctor's identity, it generates a unique token for the doctor. This token includes the doctor's identity, permissions, and session information. When the doctor wants to access a patient's medical record, this request is sent to a smart contract. The smart contract evaluates the doctor's token and access request. The token indicates the doctor's role within the hospital and the corresponding permissions. The smart contract checks the doctor's permissions and determines whether they have access to the patient's medical record. If the doctor's permissions are appropriate for accessing this data, the smart contract grants access, and the doctor can view the patient's medical record. Data sharing and collaboration between hospital domains are conducted through a consortium blockchain network. This consortium blockchain network ensures secure and transparent data exchange between Hospitals A and B. Data security and access control are maintained through smart contracts, and patients' medical records and other important data are

stored on a decentralized IPFS (Inter Planetary File System). When inter-hospital data sharing is necessary, such as when a doctor from Hospital A needs to access a patient's medical record at Hospital B, this request is evaluated through smart contracts. The smart contract at Hospital A checks the doctor's permissions and communicates with the smart contract at Hospital B to provide access. In this way, the doctor can securely access the patient's medical record at Hospital B.

Electronic Health Records are used in the healthcare industry to store patient information, clinical notes, laboratory results, medical scans, billing data, medical history, and insurance details. Therefore, protecting patients' anonymity is critical, and a breach of confidentiality can lead to major security problems. In this context, various security protocols have been developed to ensure data confidentiality and patient anonymity. However, solutions have been offered to overcome the shortcomings seen in some two-factor authentication schemes [FU4, FU5]. Current authentication protocols are vulnerable to guessing attacks, data manipulation, and denial of service attacks.

Recently, Tan et al. [FU6] proposed a secure authentication technique based on Elliptic Curve Cryptography for Medical Information Systems. Yoo and Yoon [FU7], on the other hand, introduced a mutual authentication mechanism to protect user privacy. However, these authentication procedures were completely ineffective in ensuring user anonymity and untraceability. Lin and Fan [FU8] developed a privacy-preserving protocol for Medical Information Systems to address the weaknesses of these protocols, but their approach remained vulnerable to password-guessing attacks.

In recent years, the demand for blockchain privacy and security has increased in the healthcare industry due to the anonymity, autonomy, encryption, and immutability features provided by blockchain technology. Researchers have provided solutions to many challenges considering the great benefits of blockchain in medical information systems [FU9]. Kuo et al. [FU10] have addressed many barriers to blockchain adoption in the healthcare industry and provided solutions. Zhang et al. [FU11] proposed a secure blockchain-based authentication method to protect patients' health data. Fan et al. [FU12] introduced Medblock, a blockchain-based system for processing patient data. Increases health record security through authentication and symmetric encryption. Medblock uses unified and permissioned Blockchain networks for privacy and security. Key features include secure data processing and storage that ensures data integrity. Medblock offers advanced security, privacy, and data integrity to manage health records [FU13].

In summary, Security and privacy shortcomings of traditional Electronic Health Records systems:

- Prediction attacks, manipulation and denial of service attacks
- Inability to fully ensure user anonymity and untraceability
- Vulnerability to password-guessing attacks

Blockchain-based solutions:

- Data integrity and immutability with distributed ledger technology
- Resistance to malfunctions and attacks with a decentralized structure
- Encryption and anonymity with Public Key Infrastructure (PKI)
- Permissioned blockchain networks to keep patient data private and secure
- Authentication and symmetric encryption integration with systems such as Medblock

Therefore, blockchain technology has the potential to increase data privacy and security in health information systems significantly.

3.1.2 Functional Requirements

Functional requirements define the basic functions that the system must perform.

- **Access Control Policies Management:**
    - Identification and Storage: Defining access control policies for health data and storing them securely.
    - Updating and Monitoring: Providing mechanisms for updating and monitoring policies, for example, patient consent and physician access authorizations.

These steps are illustrated in Figure 3.

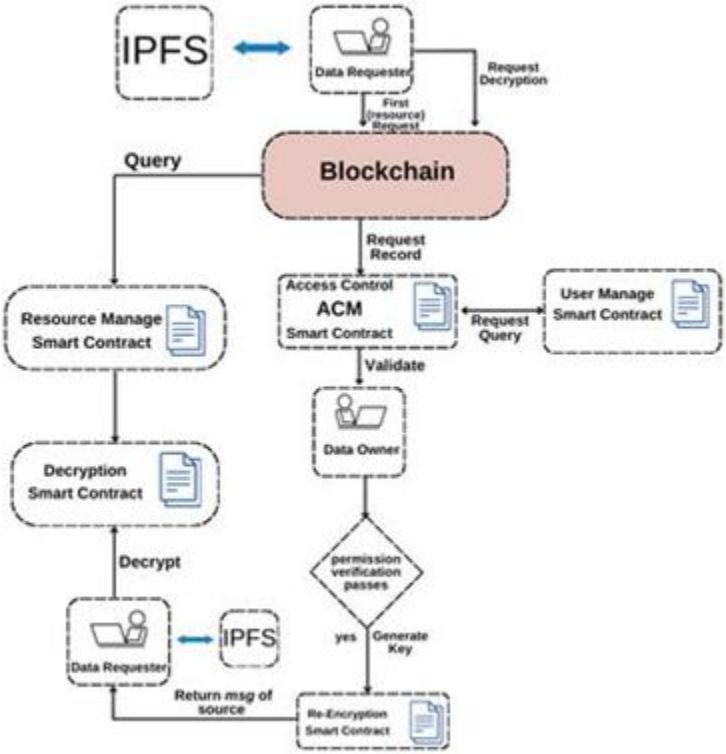

Figure 3**.** Access Control Management and access control policies

- **Identity Management**
  - Authentication: The use of cryptographic techniques for secure authentication of patients and healthcare professionals.
  - Authorization: Verifying and managing access authorizations to health data, for example, ensuring that only relevant physicians have access to patient records.
- **Data Access Monitoring**
  - Access Records: Recording and making traceable all health data access transactions.
  - Monitoring and Auditing: Regular monitoring and auditing of access records, especially for sensitive health data.
- **Data Sharing**
  - Secure Data Transfer: Between different healthcare institutions and interested parties. To ensure secure sharing of health data.
  - Transparency: Data-sharing processes must be monitored by all stakeholders, especially in cases requiring patient consent.
  - Steps of Data Sharing Mechanism:

1. Upload resource and policies: Data Owner à Blockchain
2. Format policies and store them in the ledger: Policy Manage Smart Contract
3. Upload resources to IPFS: Blockchain through the Software Interface of IPFS
4. Return IPFS hash: Resource Manage Smart Contract through Software Interface of IPFS
5. Query of User's Public key of Data Owner: Resource Manage Smart Contract of User Manage Smart Contract
6. Encrypt hash and record user to the ledger: Resource Manage Smart Contract à Blockchain

- **Consensus Mechanisms**
  - Validation: Using distributed consensus algorithms to verify the accuracy and integrity of health data access and sharing transactions.

3.1.3 Non-Functional Requirements

Non-functional requirements determine the characteristics of the system, such as performance, security, usability and compatibility.

- **Security**
  - Data Privacy: Protecting sensitive health data against unauthorized access.
  - Data Integrity: Health data on the blockchain is immutable meaning unchangeable and indelible.
- **Performance**
  - Scalability: The system must be able to scale efficiently with increasing number of patients and amount of data.
  - Speed: Performing access and verification operations quickly and efficiently.
- **Availability**
  - Accessibility: Ease of accessing health data securely.
  - User Experience: User-friendly interfaces should be provided so that users can easily perform operations on the system.
- **Durability**
  - Fault Tolerance: The system must be resilient to possible hardware or software faults.
  - Uninterrupted Operation: Continuous access should be provided against service interruptions, especially in emergency health situations.
- **Compatibility**
  - Regulations: The system must comply with relevant legal and regulatory requirements in the healthcare industry.

- Standards: Must be developed in accordance with international health data standards.

3.1.4 Component Internal Architecture

Security requirements in medical information system

- **Security:**
  - Data Privacy: Ensuring that patient data is accessible only to authorized persons. This involves encrypting data both in storage and during transmission.
  - Access Control: Limiting access based on user role and requirement using role-based access control (RBAC) and multi-factor authentication (MFA).
- **Integrity:**
  - Data Accuracy: Maintaining the accuracy and consistency of data throughout its lifecycle. Using cryptographic hash functions to detect unauthorized changes.
  - Audit Trails: Keeping detailed records of all access and modification activities. This helps in detecting and reviewing unauthorized changes.
- **Availability:**
  - System Reliability: Ensuring that the system is available and operational when needed. Using backup systems and backup solutions to minimize downtime.
  - Disaster Recovery: Having a comprehensive disaster recovery plan to quickly restore services and data in the event of system failures or natural disasters.
- **Authentication:**
  - User Authentication: Verifying the identity of users before granting access to the system. This includes the use of passwords, biometric authentication, and smart cards.
  - Mutual Authentication: Ensuring that both the user and the system authenticate each other, thus preventing man-in-the-middle attacks.
- **Authorization:**
  - Permission Management: Defining and managing user permissions, ensuring that individuals only access data necessary for their role.
  - Principle of Least Privilege: Giving users the minimum level of access necessary to perform their tasks.
- **Auditability:**
  - Comprehensive Record Keeping: Keeping detailed records of all system activities, including entries, data access, and changes.
  - Regular Audits: Conduct regular security audits to review logs and identify potential security breaches or policy violations.
- **Undeniability:**

    - Digital Signatures: Using digital signatures to ensure that actions and transactions within the system can be attributed to specific individuals, thus preventing actions from being denied.
- **Anonymity and Pseudonymity:**
    - Data Anonymization: Data anonymization to protect patient identities in data sets used for research or other non-clinical purposes.
    - Use of Pseudonyms: Protecting confidentiality by changing identity information with pseudonyms and ensuring that identities remain confidential when analyzing data.
- **Security Protocols:**
    - Encryption: Using strong encryption algorithms for data storage and transmission.
    - Secure Communication Channels: Using secure communication protocols such as HTTPS, SSL/TLS to protect data transmission.
- **Compatibility:**
    - Legal and Regulatory Compliance: Ensuring that the system complies with applicable laws and regulations, such as HIPAA in the US or GDPR in Europe.
    - Regular Updates: Keeping the system updated with the latest security patches and updates to reduce vulnerabilities.

Solution Examples in Medical Information Systems:

- **Blockchain Integration:**
    - Data Integrity and Immutability: Blockchain being inherently immutable ensures that data cannot be altered once recorded and provides a reliable audit trail.
    - Distributed Access Control: Distributed ledger technology manages access control without a central authority, increasing security and reducing single points of failure.
- **Advanced Cryptographic Techniques:**
    - Elliptic Curve Cryptography (ECC): Provides strong encryption with shorter keys, ensuring effective and secure encryption of data.
    - Homomorphic Encryption: Allows data to be processed without being decrypted, protecting data confidentiality even during the process.
- **Federated and Permissioned Blockchain Networks:**
    - Controlled Access: Federated networks involve multiple stakeholders who control the blockchain together, so no single entity has complete control.
    - Privacy-Preserving Techniques: Permissioned blockchains can implement privacy-preserving protocols to ensure that only authorized participants can access them.

Implementation of these security requirements plays an important role in ensuring the security of medical information systems and protecting sensitive patient data against various threats and vulnerabilities.

Proposed decentralized authentication protocol for Medical Information Systems:

The proposed protocol consists of initialization phase, user registration phase, mutual authentication.

A. **Initialization phase:** A secret key is created between Hospitals and the Blockchain Network using the Diffie-Hellman Key Exchange Algorithm.

B. **User registration phase**

Steps of User Registration:

1. Register identify of user
2. Get local user information
3. User identify validation
4. Generate certificates for users
5. Identity authentication
6. User register
7. Record User and public key of user to ledger
8. Return private key of user

If a patient or doctor wishes to register with a medical server connected to the Blockchain Network, they must do so using a secure communication channel. The user registration may include the following process:

- **User's ID and Password Selection:** First, the patient or doctor identifies the patient or doctor, chooses a password, and generates a random number. This random number provides an additional layer of security in the authentication process.
- **Sending the Registration Request:** The user's device prepares a registration request using these credentials (ID, password, and random number). This request is sent to the medical server connected to the Blockchain network via a secure communication channel. A secure communication channel is a method where data security is ensured, usually by using encryption technologies.
- **Registration Process on Blockchain Medical Server:** The medical provider takes certain security measures after receiving the registration request:

  - Counter Calculation and Initialization: Calculates and initializes a counter to prevent replay attacks. This counter is unique for each registration request and is used once.
  - Registration to Blockchain: Writes the user's registration information to the Blockchain. Blockchain works as a decentralized database and ensures that records are stored in an immutable and secure manner. For each registration transaction, a unique transaction ID is created in the Blockchain.
- **Sending Registration Response Message:** The medical server prepares a response message indicating that the registration process was completed successfully and sends it back to the user over a secure channel.
- **User Device Performing Calculation:** The user device makes the necessary calculations using the response message it receives. These calculations are necessary to authenticate the user and provide secure access.
- **Protection of Access Information:** Finally, the user device securely stores the information necessary to access services from the medical provider. This information will be used for future user identity verification and authorization processes. This process ensures that user credentials are securely stored and verified while incorporating additional security measures to prevent replay attacks. Blockchain technology increases the security of medical data by ensuring that this process is carried out reliably and transparently.

C. Mutual authentication

Mutual authentication involves both parties verifying each other's identity to ensure secure communication between the user and the medical provider. This process is accomplished by exchanging and verifying authentication messages. This process is as follows:

- **Exchanging Authentication Messages:**
  - Start: The user (patient or doctor) sends an authentication request to the medical server. This request usually includes the user's credentials and a security token.
  - Server Response: The medical server receives the credentials and security token from the user and initiates the verification process.
- **Authentication and Accounting:**
  - User Verification: The medical server verifies the user's credentials by comparing them with its database. During this verification process, the user's password, biometric data, or other authentication information may be used.
  - Result: If the user successfully passes the authentication process, the medical server calculates a session key after authenticating the user. This session key is used to encrypt all subsequent communications.
- **Authenticating the Server:**

  - Server Information: The medical server sends its identity and verification information to the user so that the user can verify it.
  - User Authentication: The user checks the information he receives to authenticate the server. This is usually accomplished with the server's certificate or digital signature.
- **Acceptance of Session Key:**
  - Successful Authentication: If both the user and the medical server successfully authenticate each other, the user accepts the session key from the server.
  - Encrypted Communication: This session key is used to encrypt all subsequent communications, thus ensuring data security.
- **Rejecting the Authentication Request:**
  - Failed Authentication: If any party in the authentication process fails to authenticate the other party, the authentication request is rejected and the connection is terminated.
- **Authentication Protocols:** Various authentication protocols can be used to accomplish this process:
  - SSL/TLS: These are widely used protocols for secure data transmission and authentication.
  - OAuth: Provides secure authorization and authentication for third-party applications.
  - Elliptic Curve Cryptography (ECC): Used for strong encryption and authentication.
- **Blockchain Usage:** Blockchain technology can make the mutual authentication process more secure and transparent. Here are some advantages:
  - Immutability: Since data on the blockchain is immutable, authentication processes are reliably recorded.
  - Transparency and Traceability: All authentication transactions can be tracked and verified on the blockchain.
  - Secure Key Management: Blockchain enables the secure creation and management of session keys.

These methods protect the privacy and data security of both patients and doctors by providing a secure and reliable authentication process in medical information systems.

3.2 Component 2- Multi-agent and Self-aware Collaborative Intrusion Detection Systems

3.2.1 Overview of the Multi-agent and Self-aware Collaborative Intrusion Detection Systems

The envisioned framework includes the development of a multi-agent and self-aware collaborative *intrusion detection technique* that will be responsible for monitoring the network traffic for potential security events (**monitoring/detection phase**). It consists of *two parallel modules*, (1) semi-asynchronous federated learning and (2) blockchain technology. After the monitoring/detection phase, there is the **mitigation phase**, which concerns methods for the coordinate response capability of the self-aware system in case of intrusion that could affect the whole environment among different infrastructures. It will consider autonomous conceptual primaries who are responsible for protecting specific parts of the system, as well as *supervisor agents* who coordinate with each other for collaborative response. The coordination among the agents will follow the Multi-agent Systems principles where agents are autonomous and capable of performing specific actions in dynamic operational decentralized and distributed environments.

3.2.2 Functional Requirements

The functional requirements for a multi-agent and self-aware collaborative intrusion detection system can be outlined as follows:

- **Monitoring:** The system should be capable of monitoring various components of the network, such as traffic, system logs, file integrity, and user activities, to detect potential threats.
- **Threat Detection:** The system should be able to identify and classify different types of threats, including known and unknown attacks, using techniques like signature-based detection, anomaly detection, and machine learning models.
- **Collaboration and Information Sharing:** Agents should be able to communicate and share information about detected threats, learned models, and system state to enhance overall detection capabilities.
- **Adaptation and Learning:** The system should be capable of adapting to new and evolving threats by continuously updating its detection models through cooperative learning and knowledge sharing among agents.
- **Reconfiguration and Self-Healing:** In case of agent failure or compromise, the system should be able to dynamically reconfigure and reorganize the remaining agents to maintain functionality and mitigate the impact.

- **Response and Mitigation:** The system should be capable of initiating appropriate responses to detected threats, such as generating alerts, blocking malicious traffic, or initiating incident response procedures.

3.2.3 Non-Functional Requirements

The functional requirements for a multi-agent and self-aware collaborative intrusion detection system can be outlined as follows:

- **Scalability:** The system should be able to scale horizontally to accommodate larger networks and increasing traffic volumes without compromising performance.
- **Fault Tolerance:** The system should be resilient to agent failures or compromises, ensuring that the overall functionality is not significantly impacted.
- **Performance and Efficiency:** The system should have minimal overhead and impact on network performance, while still providing timely threat detection and response.
- **Security:** The system itself should be secure, with measures in place to prevent unauthorized access, ensure secure communication among agents, and protect the integrity of learned models and shared information.
- **Interoperability:** The system should be able to integrate with existing security infrastructure, such as firewalls, security information and event management (SIEM) systems, and other security tools.
- **Configurability and Customization:** The system should be configurable to adapt to specific organizational requirements, network topologies, and security policies.
- **Usability and Manageability:** The system should provide intuitive interfaces and management tools for configuring, monitoring, and analyzing the system's performance and detected threats.
- **Privacy and Compliance:** The system should comply with relevant privacy regulations and organizational policies regarding data collection, processing, and storage.

These requirements serve as a guideline for designing and developing a robust, efficient, and adaptable multi-agent and self-aware collaborative intrusion detection system that can effectively protect against a wide range of cyber threats.

3.2.4 Component Internal Architecture

The envisioned framework consists of *two parallel modules*, (1) **semi-asynchronous federated learning** and (2) **blockchain technology**.

(1) The designed semi-asynchronous algorithm will find the optimal number of workers that will potentially save the training time in the communication round of non-IID data, and

second, it aims to improve overall accuracy with a high intrusion detection rate and low false alarm rate. Alternatively, the algorithm will find a balance between stale and up-to-date model updates to avoid biased global model generation, using novel aggregation clustering methods. Ultimately, the FL solution will be able to handle the heterogeneity presented by agents in terms of computational power, communication speed, and non-IID data

(2) Afterward, a new blockchain architecture will be designed to store and validate the model parameters with less computational and communication costs. Further, based on the gathered intelligence, a self-aware collaborative IDS will be designed to detect unseen attacks in distributed and decentralized smart infrastructure.

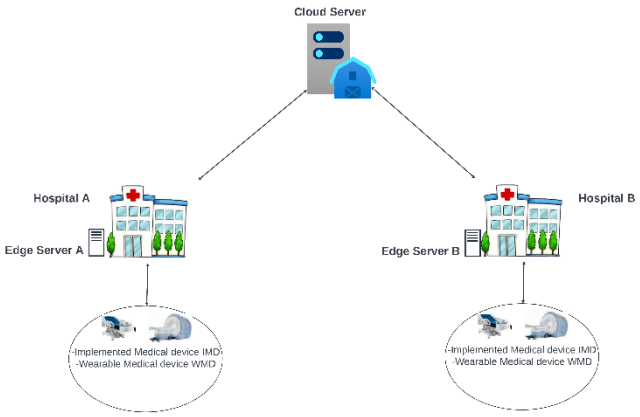

Figure 4. Cloud and edge servers

Figure 4 illustrates the architecture of the edge servers and the cloud server that intervene in the FL process. The description of components within each of them is described in the following.

- **Edge Server roles:**
  - Data Collector: a component for collecting network flows and logs
  - Data Processor: a component for pre-processing the data
  - Training and real-time monitoring: a component for IDS deployment and training
  - Resource allocation: a component for bandwidth allocation for communication
- **Cloud Server roles:**
  - Aggregator: aggregate received local models

- Orchestrator: diffusing the global model and the ability to store the local models. This component will be responsible for the training orchestration in FL, i.e., semi-asynchronous FL. For example, it will decide when to synchronize the clients and it will be responsible for coordinating the training. Objective, minimize FL training duration and mitigate the staleness problem.
- Client Selector: a client selection module.
- Training Configurator: Assign the training configuration for the clients

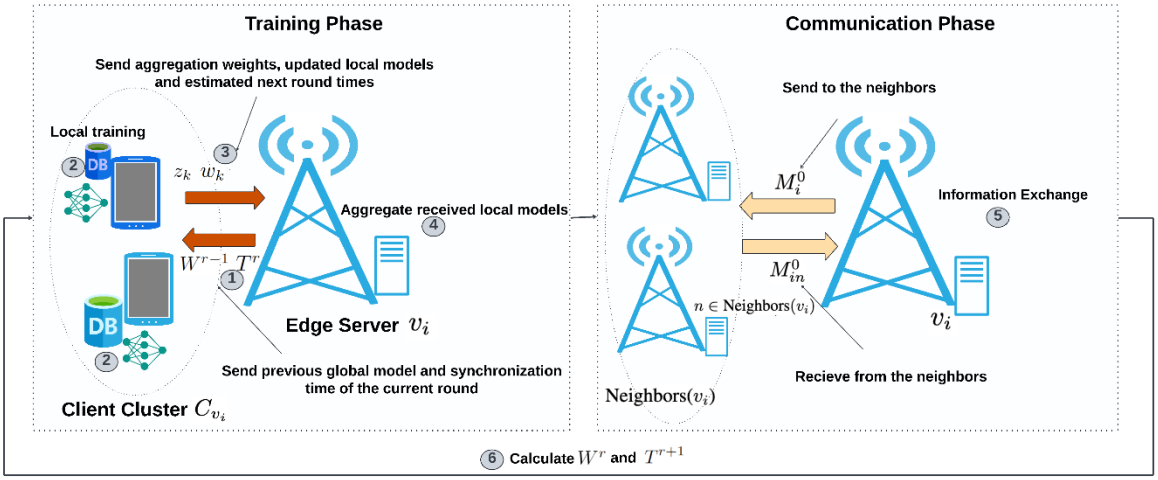

Figure 5: Training and communication phases

Figure 5 illustrates the training and the communication phases. Each FL global round includes multiple steps. At the beginning of each global round *r,* edge servers trigger the training phase by dispatching the prevailing global model *W^{r-1}* along with the synchronization time *T^r* to their assigned clients (step 1). Then each device *k* starts its local training using its private dataset *D_k*. They continue to train until the deadline *T^r* is met, or they have converged before it (step 2). After the deadline, each client *k* estimates, the time required to achieve convergence for the subsequent global round, then communicates its participation weight *z_k*, local model update *w_k* and estimated next round time *S_k^{r+1}* to their respective edge servers *v_i* (step 3), who will collect this information, aggregate the local updates received and saves the new sub-global model *W^r_{v_i}* together with the set of estimated next round time for each client (step 4). Following this, edge servers trigger the communication phase (step 5), they exchange information through message-passing to finally compute a unified global model *W^r* and a next-round synchronization point *T^{r+1}* (step 6), reflecting the insights of all participating devices. An example of dynamic synchronization points are shown in Figure 6.

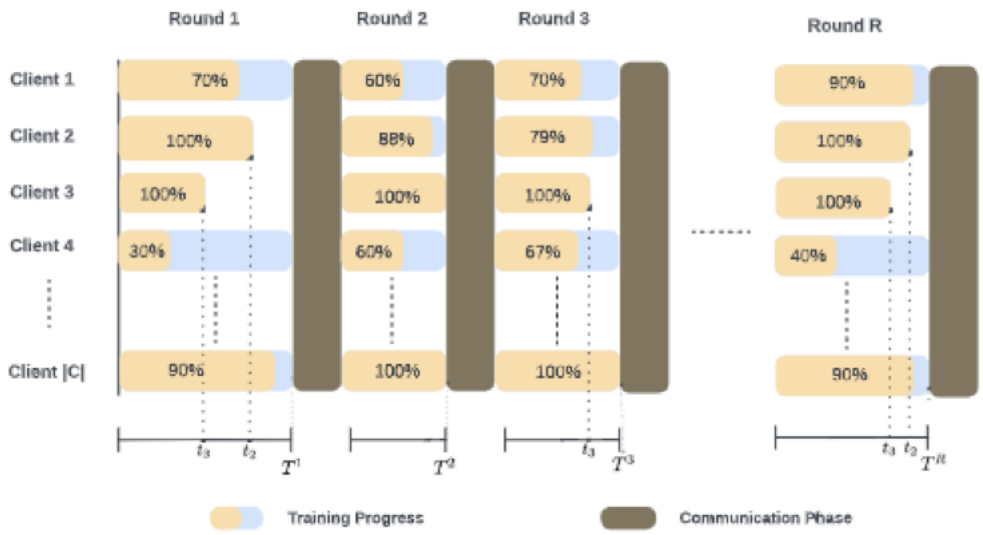

Figure 6: Example of dynamic synchronization points

3.3 Component 3- Dynamic Risk Management

3.3.1 Overview of the Dynamic Risk Management

Our aim in this part is the development of a risk management framework that defines the main steps for risk analysis, management, and treatment services specifically related to the distributed and decentralized system, taking into account individual and cascading risks. This framework should be general enough, but it should possibly be specifically tailored to the case of distributed and decentralized systems, and it should provide advanced vulnerability exploitability prediction services using a machine learning model. The framework has to be evidence-based and dynamic due to the consideration of the evolving system and threat context and consider interdependencies among the assets for the propagation of vulnerabilities.

3.3.2 Functional Requirements

The main functional requirements of the dynamic risk management framework are as follows:

- **FR1:** The framework has to deal with assets, threats, security domains, security dimensions, security controls and the calculation of Key Risk Indicators, specifically for distributed and decentralized systems, but also for other possible related types of systems (e.g. critical infrastructures, CPS systems, Big Data systems, etc).

- **FR2:** The framework has to support de definition of catalogues (assets, threats, controls, etc.) based on standards, guidelines and the most accepted specifications, that can be adapted to specific contexts.
- **FR3:** The framework has to define an architecture that allows the development of several inter-connected instances of risk analysis for complete or partial distributed and decentralized systems, offering inheritance and association mechanisms to dynamically propagate risk incidents and indicators.
- **FR4:** The framework has to allow the reutilization of specific risk catalogues to different risk analysis instances or projects.
- **FR5:** The framework has to provide risk analysis instances that can dynamically and quickly evolve according to the security incidents once they happen, adjusting the level of risk and the necessary controls, triggering the necessary alerts, and propagating the necessary risk adjustments to other associated instances. Therefore, this way of data propagation must provide the possibility of collaboration within an ecosystem of enterprises and/or organizations in order to protect themselves from security risks.
- **FR6:** The framework has to provide the appropriate interfaces both for receiving risk inputs in real time (not only threats over the already known vulnerabilities stored in public CVEs, but also other security incidents or events) and generating outputs with risk indicators that can be useful to other decision support systems.
- **FR7:** A set of Key Risk Indicators have to be defined to manage security risk of distributed and decentralized systems, their dependencies and relationships.
- **FR8:** The framework has to be supported by a cloud-based software that allows automatize its main processes.

3.3.3 Non-Functional Requirements

Frequently, Functional Requirements that lack sufficient detail are considered Non-Functional; however, once they are properly developed, these can be easily interpreted as either Functional or Quality Requirements [UCLM13]. One of the most widely accepted taxonomies of software quality requirements defines the following list [UCLM14]: Availability, Efficiency, Flexibility, Integrity, Interoperability, Reliability, Robustness, Usability, Maintainability, Portability, Reusability, and Testability. These quality requirements are important for our Dynamic Risk Management system in certain respects, so below we provide a specific interpretation and limitation for each one:

- **Availability:** The cloud-based software has to be available 24x7 for updating the happened risk events and to generate risk results and trigger security alerts.

- **Efficiency:** The algorithms and processes applied in the risk analysis and management can be highly demanding from the time and space perspective, so innovations have to be applied to obtain efficient algorithms.
- **Flexibility:** The framework has to be flexible enough to be applied not only to distributed and decentralized systems (parts or complete) but also to other systems and be able to interact flexibly with each other.
- **Integrity:** The risk information that is stored and managed within this framework is highly sensitive. So, mechanisms to ensure its integrity and its confidentiality are necessary.
- Interoperability: The risk framework has to interoperate with other systems and persons, so, the necessary interfaces have to be included to facilitate the incoming and outgoing communication.
- **Reliability:** A risk analysis and management framework necessarily has to be reliable. Mechanisms supporting the explainability in the risk related decisions are necessary.
- **Robustness:** The framework will ensure that the critical risk decisions are supervised by human stakeholders, but supported by the software.
- **Usability:** One key factor of this solution is approach it to any company or organization, regardless of their security budget. This fact requests a sustainable and easy to use solution, and that optimize the use of personal and economic resources. So the solution has to be focused on a high usability.
- **Maintainability:** According to the highly dynamic character of the world of security, vulnerabilities and threats, the framework has to be prepared to be easily adapted to these changes.
- **Portability:** The automatic support of the risk framework has to be cloud and web based to ensure the portability.
- **Reusability:** The software and the data defined within the adaptative risk catalogs have to be fully reusable from a risk analysis instance to any other.
- **Testability:** In order to be able to test the tool and the framework, simulations can be carried out, and a rich set of balance score cards will allow to visualize and analyze the most important risk indicators.

3.3.4 Component Internal Architecture

The envisioned framework, which we have named MARISMA, consists of two main components. The first component is the ”MARISMA Method”, with the risk analysis and management process as well as the information model or risk meta-pattern (reusable and applicable in different contexts) and the specific patterns which are created for specific sectors using the elements of the risk meta-pattern. Within this part, a specific template for risk analysis and risk management in DSS will be developed as a core component. Finally, the second component is ”Automatic

Support", where the eMARISMA software tool is used to support the risk meta-pattern and the process mentioned above and to implement the specific patterns. Within this basic tool, specific modules will be implemented to support monitoring, analysis and risk management adapted to the specific characteristics of the DDS. Figure 7 provides an overview of this framework:

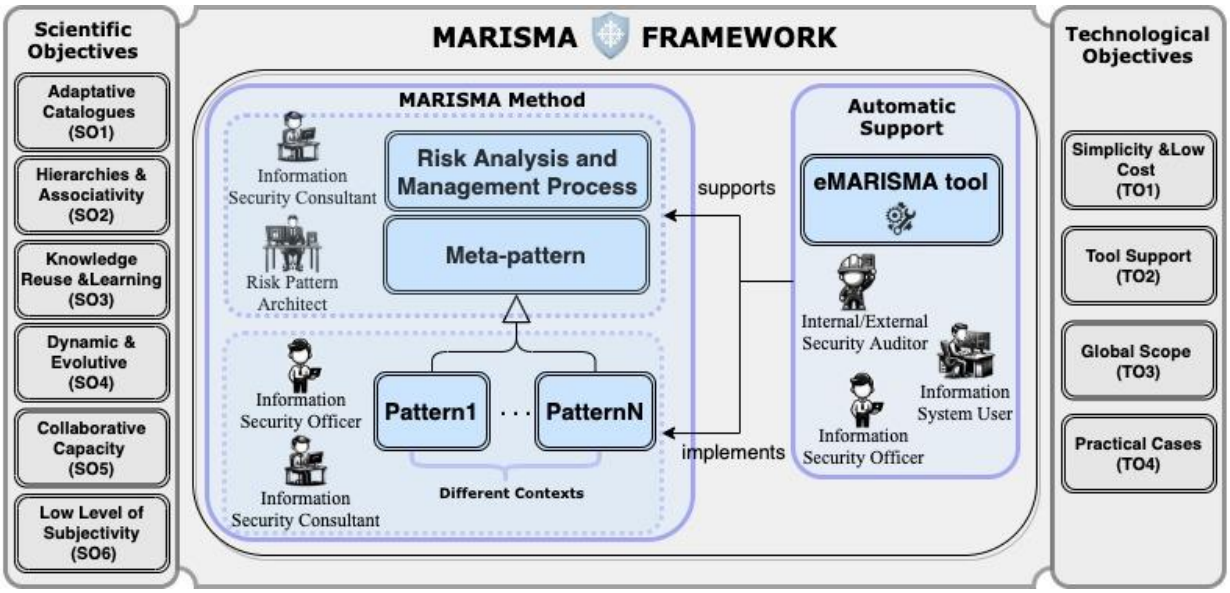

Figure 7**:** MARISMA Framework

The risk meta-pattern structure outlines the elements and relationships necessary for data modeling to conduct risk assessment and management in any organization or technological environment. It primarily defines the required structure of Controls, Assets, and Threats (CAT), along with the relationships that detail the semantic aspects of each pattern (intra-pattern relationships) and the mechanisms that enable the establishment of pattern hierarchies (inter-pattern relationships) through the reuse and inheritance of control, asset, and threat components. New patterns can be generated to extend the meta-pattern structure, creating target patterns tailored to specific contexts, such as a particular sector (even for business process models [UCLM15]), company size, system type (e.g., Distributed and Decentralized Systems), and specific technology (e.g., Big Data) [UCLM16, UCLM17]. This approach leverages knowledge from previous implementations to enhance and expedite the development of new risk assessments and management in different contexts. Furthermore, a global pattern can be applied to an organization, with more specific patterns for different parts of the organization, such as divisions, departments, and technologies. This enables the establishment of relationships and dependencies between elements of different risk patterns, allowing for the representation and inheritance of hierarchical risk structures, as well as the associativity between various implementations.

Figure 8 shows how, in our method, the MARISMA meta-pattern, specific patterns, and intra-pattern relationships are positioned. For its representation, it uses the meta-object facility (MOF) standard from the Object Management Group [UCLM18], which provides a framework for defining metamodels organised into four levels of abstraction (from M3 to M0).

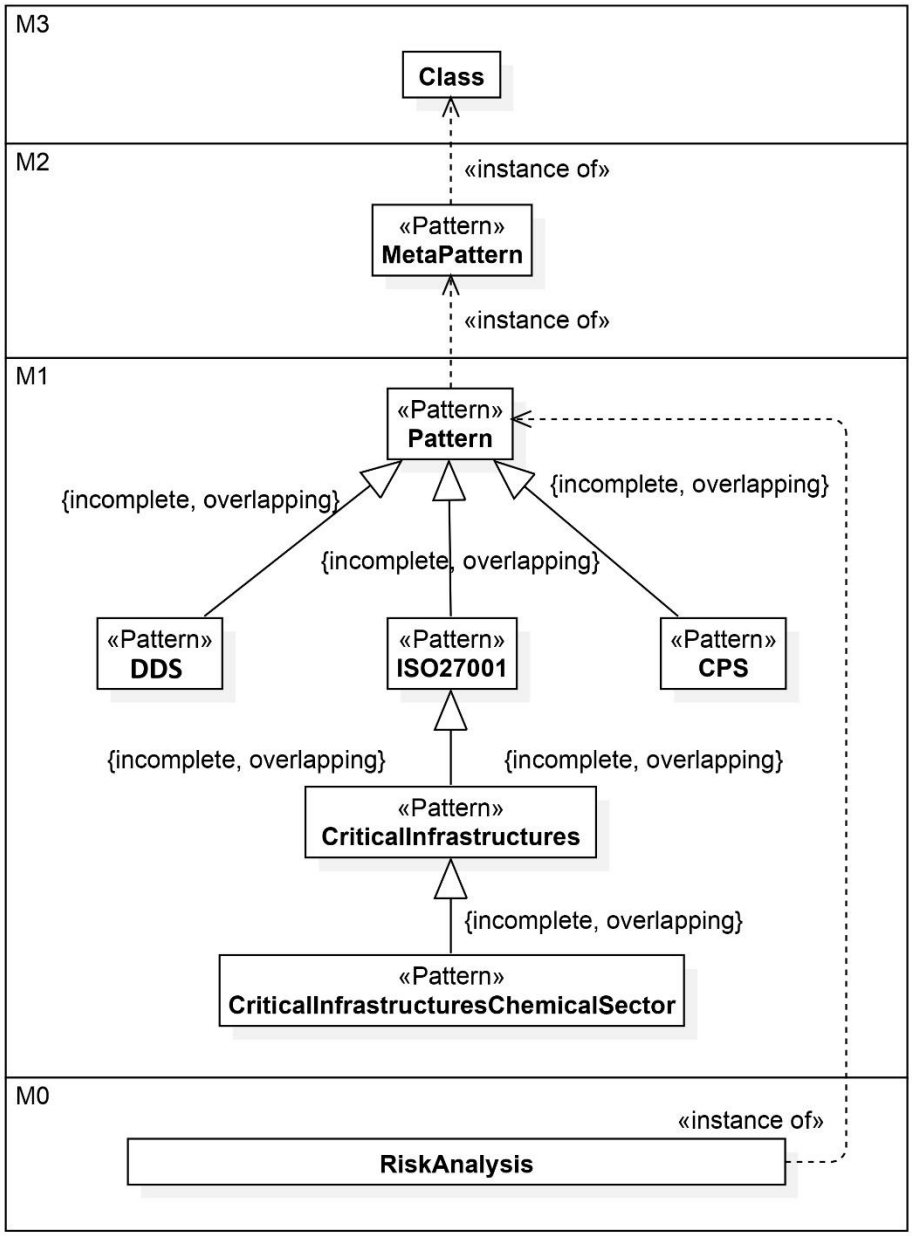

Figure 8**:** Pattern inheritance and instantiation

- M3 represents the highest level, the meta-metamodel, where MOF defines itself and describes how metamodels should be constructed.
- At the M2 level, the metamodels that describe the structure and rules that must be followed by the models based on them are defined. The MARISMA meta-pattern was placed at this level.
- At the M1 level, models are defined according to a specific metamodel, and this is where we place the specific patterns (And that it will support the specific pattern for DDS) defined according to the MARISMA meta-pattern.
- Finally, at the M0 level, we identified concrete instances of the models when they were applied to specific analyses. Within the project, and once a prototype of the system is available, each of the implementations of the framework in hospital environments will represent a differentiated risk analysis (although supported by the same architecture defined by the DDS pattern).

This hierarchical structure ensures a clear and organized approach to modeling, facilitating the use of the MARISMA meta-pattern in various contexts, which provides a mechanism for investigating, designing and implementing a specific risk pattern for DDS infrastructures.

A more detailed view of risk metapattern is presented in Figure 9. This shows a UML model which includes meta-classes that define the main elements, as well as their interrelationships and interdependencies. The set of components outlined below can be clearly identified in this model.

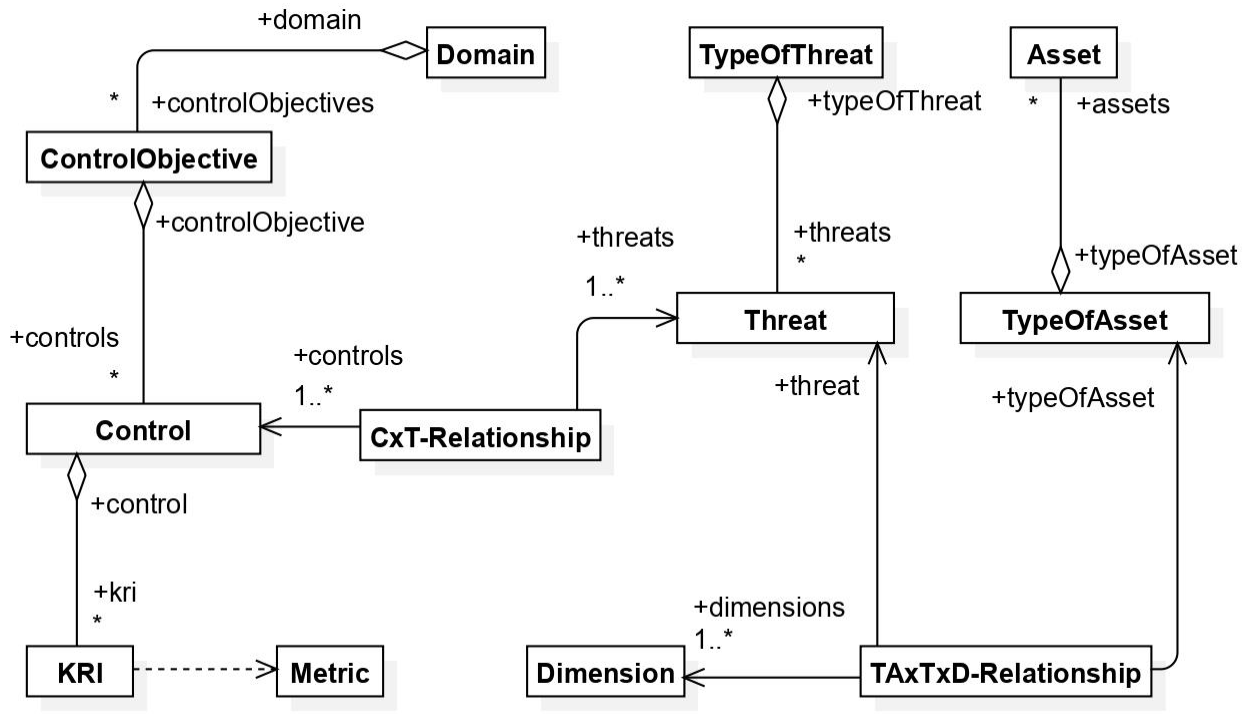

Figure 9: Meta-pattern CAT (detailed view)

- **Controls:** Using this structure, a security pattern architect can define the safeguards and security measures to be implemented in a specific risk analysis to control the impact of threats that may affect an organisation's assets [UCLM19]. The structure comprises an aggregation hierarchy that groups security controls into targets to be controlled, which are then grouped into normative domains to be managed. MARISMA considers vulnerabilities as the absence of appropriate security controls. Finally, security controls are linked to key risk indicators (KRIs), which are informed by metrics that facilitate knowledge and improvements in successive implementations. Within this component, the DDS hierarchy of Domains, Control Objectives and Controls is designed and maintained.

- **Assets:** This structure allows an organisation's assets to be defined and categorised. Assets, which can be grouped by type, are also associated with different dimensions based on potential threats to elements such as confidentiality, integrity, and availability. Consequently, assets can be analysed from multiple perspectives (dimensions). Within this component, a taxonomy of asset types specific to DSD will be designed and derived.

- **Threats:** This structure allows the identified threats to be defined and organised into groups, facilitating improved reuse and threat management. These threats are linked to

assets and controls, which are defined by certain components in the form of matrices. This component will define and develop a catalogue of threat types and threats that may affect and potentially cause security incidents within the DDS.

- Intra-pattern relations:
    - Type of Asset x Threat x Dimension (TAxTxD) relations: Using this matrix, the relationships between different types of assets, potential threats, and affected dimensions can be specified. These are considered intra-pattern relationships, which help optimise risk analysis by preventing incorrect tuples that could unnecessarily complicate the process, as not all assets are affected by every threat or related to all possible security dimensions [UCLM20].

    - Control x Threat (CxT) relations: This matrix allows for the specification of control-threat interdependencies. Controls are defined to prevent or mitigate the effects of threats, whereas the absence of security controls related to a threat indicates an incomplete protection scenario. The absence of threats related to a control might suggest that the control can be disregarded in this particular risk analysis. This matrix, representing an intra-pattern relationship, is linked to the TAxTxD relationship to determine the controls applicable to specific types of assets.

    - Inter-pattern relations: Our approach not only allows for the definition of specific risk patterns but also facilitates the development of new risk patterns by inheriting components from previously defined patterns. This is a crucial feature of the framework, which can be leveraged to create a highly dynamic and intelligent risk analysis and management ecosystem.

Within this element of the MARISMA framework is also the set of processes of which the methodology is composed, and comprises three processes that deal with the risk analysis and management life cycle as seen in Figure 10 using (SPEM 2.0) [UCLM21], since they make the system dynamic, thus allowing it to evolve over time. These three processes are:

- The RPG Process (Risk Pattern Generator) aims to generate patterns for risk analysis, including their relationships and the knowledge acquired from various implementations. This process is responsible for defining all components of a specific risk pattern, which includes types of assets [UCLM22], controls, threats, security dimensions, and their interrelationships. Essentially, it involves creating a new pattern for a specific context. The risk pattern is developed by analysing the most appropriate standards and reference documents related to the application area and through direct inheritance from our meta-pattern or another previously developed risk pattern, leveraging the knowledge and experience gained from prior implementations. Additionally, once the pattern is fully operational, inputs in the form of security events are received to update different parameters of the risk pattern.

- The RAMG Process (Risk Analysis and Management Generator) focuses on generating risk analysis and management through the instantiation of the most suitable pattern. It also enables the definition of dynamic metrics to value assets and the risk calculation mechanisms. Once the risk analysis is generated and risks are managed through the Dynamic Risk Management (DRM) process, the identification of security events can lead to dynamic enhancements in the parameters of risk analysis and management.

- The DRM Process (Dynamic Risk Management) involves the dynamic maintenance of risk analysis through the use of matrices that interconnect various artefacts, allowing the system to recalculate risk as security incidents occur, defined metrics fail, or expert systems generate suggestions. This process can update certain parameters of risk analysis and management, such as threat probability and degradation rate, to enhance a specific case. Additionally, security events, such as threats to previously unidentified assets, may be identified, potentially leading to further modifications at the risk-pattern level. Once these modifications are incorporated into the pattern, they enhance all risk analyses and management derived from this pattern.

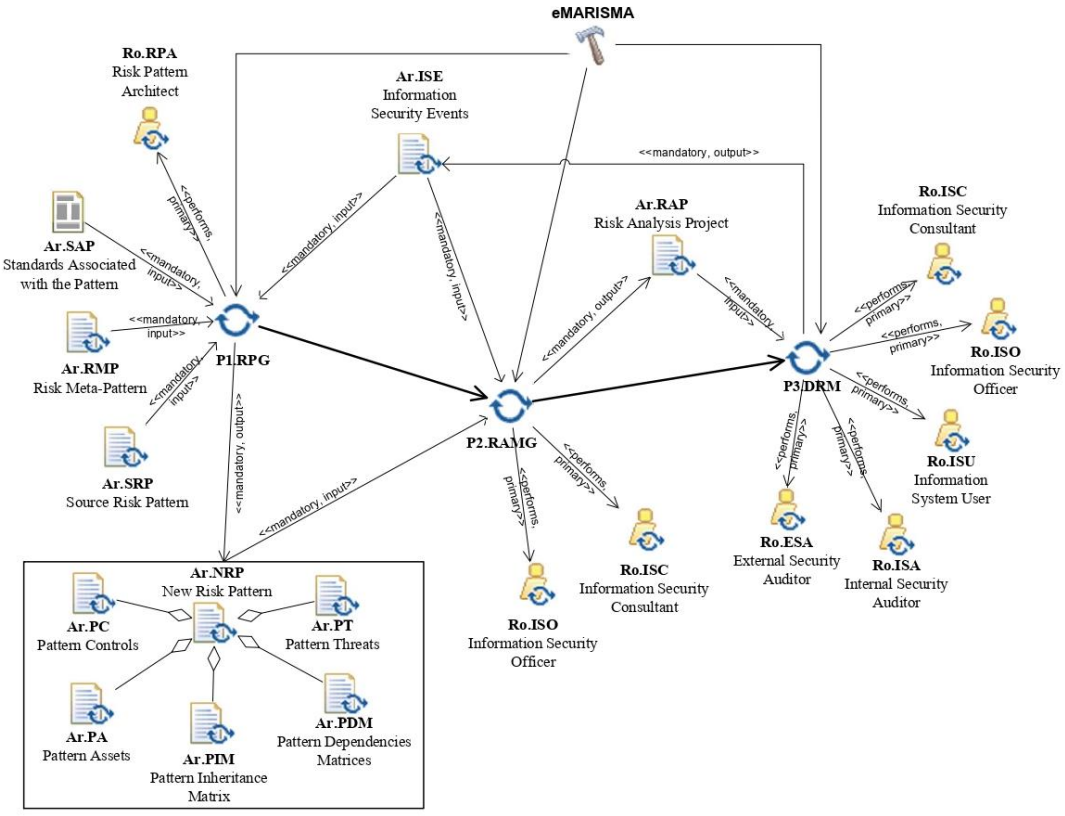

Figure 10. Framework processes overview

Thus, the pattern's structure allows new elements to be added to a given instance, enabling it to adapt and evolve; for example, new threats or controls can be incorporated into a pattern to meet changing needs. Additionally, both patterns and risk analyses can evolve dynamically in response to security events, thereby facilitating agile adaptation to changing conditions. n addition, this pattern structuring allows the creation of numerous instances that are adapted to element catalogues in specific contexts, such as technology, sector and standard, thus enabling the development of DDS-specific catalogues, control mechanisms, security incidents and risks. Moreover, the ability to evolve dynamically through the reuse of knowledge introduces the high-potential concept of collaborative security (hive-mind). This mechanism allows companies to protect each other and make their risk analysis an element that evolves not only with internal information but also draws on a global knowledge network, which allows it to fit perfectly and fulfil its potential in decentralised and distributed systems.

The second element of the MARISMA framework is the eMARISMA tool, which was developed to provide automated support for the MARISMA framework, facilitates the management of MARISMA's knowledge base, including the risk meta-pattern, hierarchy of risk patterns, and their implementation in risk projects, and simplifies the risk analysis and management process.

This tool is developed using J2EE technology and can be accessed by customers on an Internet browser from any device. This model is based on the deployment of an application on an Apache server with a servlet container called Tomcat that separates a module to manage risk patterns from a module that manages risk analysis projects for greater security, thereby protecting the analytical data of clients and guaranteeing data confidentiality. As this platform operates in disconnection mode, it requires only a connection when requesting a risk pattern update.

Although the eMARISMA tool automates the three processes of the MARISMA framework, it is designed to be extensible to support new functionalities. In this way, an extension to support, implement and manage specific DDS risk patterns will be developed, as well as an expert system embedded in the tool to predict (and subsequently propagate to network nodes) evidence-based risks (either from vulnerability reports or from security incidents reported within the DDS).

- **Inputs and outputs:**

The component will have two types of possible inputs for which the necessary communication APIs will be implemented:

1. CVE record sets: Mechanisms will be in place to read vulnerability logs and filter out vulnerabilities that may affect DDS systems. Based on these vulnerabilities, the system will generate alerts and security recommendations as a preventive mechanism.

2. Security events: The system shall support the input of security incidents reported to the system caused by typical DDS threats. Based on each reported incident, the risk level of the system shall be automatically and dynamically recalculated, and two types of output can be produced:

i. An alert indicating that a variation in the risk level has occurred, which can be propagated to the different nodes of the system. Administrators will be able to access the tool to consult in detail the variations produced through a dashboard.
ii. A pre-emptive alert when the predictive system detects that the occurrence of the threat associated with the security incident may cause a potential increase in risk, which may also be propagated to the various nodes of the monitoring and control system.

4. Di4SPDS Reference Architecture

4.1 Overview of the Di4SPDS Reference Architecture

A distributed and decentralized smart healthcare system is proposed, where multiple healthcare providers (hospitals, clinics, etc.) collaborate through a consortium blockchain. Each healthcare provider has its system, and they store their data on a private blockchain as their server, which we can call an *edge server*. Every user should be registered first to use the system. All medical devices are also registered to join the system. As it will be distributed smart healthcare, one hospital can ask for patient information from another hospital, so how will they trust them? To ensure data integrity, a consortium blockchain server called a *P2P cloud server* is introduced. When any hospital asks for any information, the consortium blockchain server verifies the originality of the data by cross-referencing it with the data stored on all registered hospitals' edge servers.

The system is fortified with advanced cryptographic techniques and leverages federated learning to maintain the highest levels of security and privacy. This robust security framework ensures that patient data remains confidential and secure, fostering trust and confidence among healthcare providers. It also allows for collaborative research and treatment planning, further enhancing the system's value.

Component 1. BSCDA (Blockchain and Smart-contract enabled Cross-Domain Authentication and Access Control Scheme) use case:

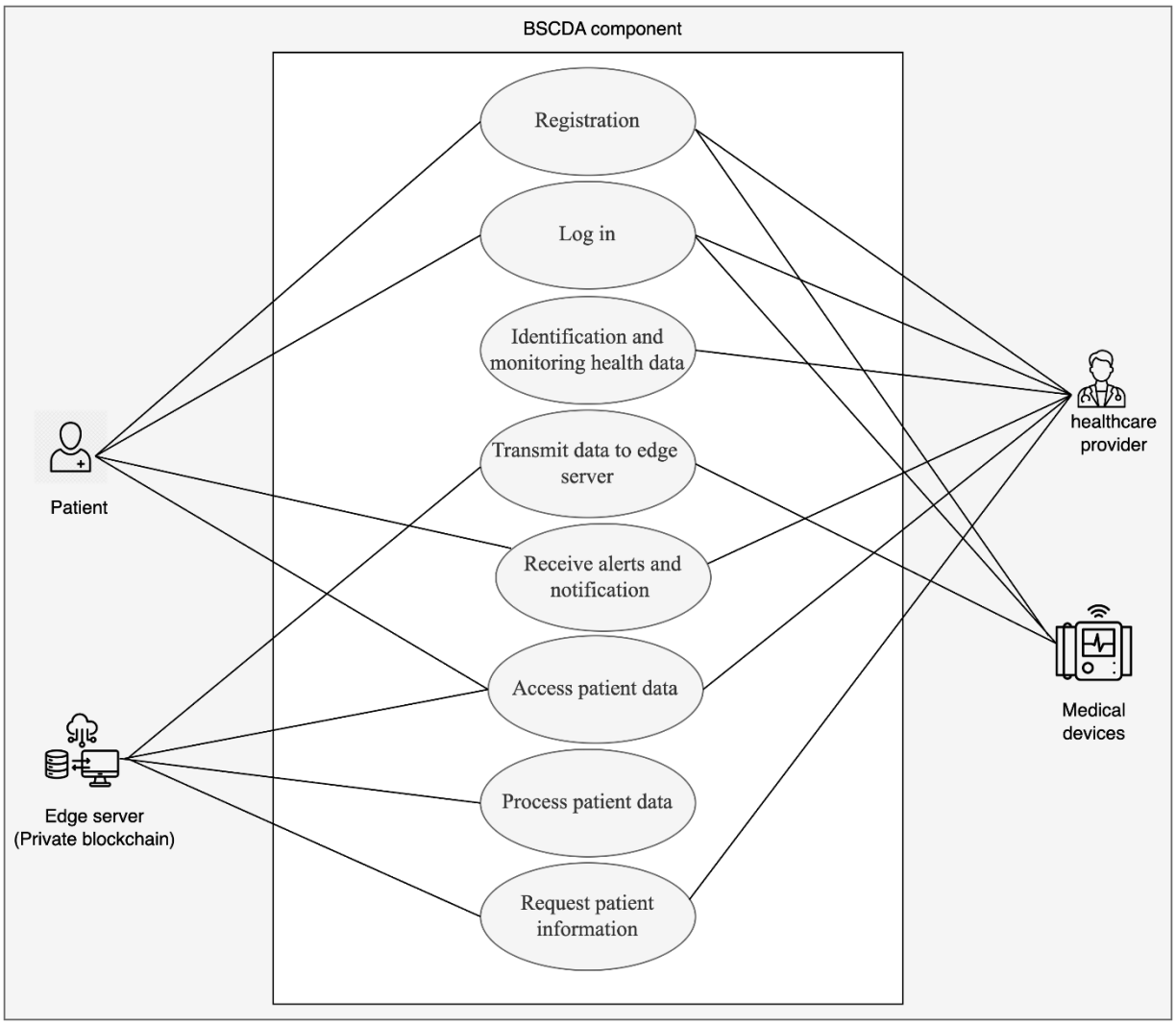

Figure 11**:** Use case diagram of BSCDA component

UC-1. Registration

Use case element	Description
Title	Device and User registration
Actors	Patient, Healthcare Provider, Medical Device
Goal	Register all the actors in the private or consortium blockchain for secure identity and access control.
Precondition	The user or device must provide valid identity information. The system must be connected to the blockchain network.
Trigger	User or device requests registration through the interface.
Function flow	• User/Device will initiate a registration request. • The system will collect identity information. • A smart contract will validate the information.

	• A new identity will be registered on the blockchain and added to the access control table. • A confirmation of successful registration message will be provided to the user/device.
Postcondition	The user or device can log in to the system

UC-2. Log in

Use case element	Description
Title	Securely log in to the system
Actors	Patient, Healthcare Provider, Medical Device
Goal	Authenticate users or device to provide access to system resources through blockchain-based smart contracts.
Precondition	The actor must be registered in the system.
Trigger	Actors attempt to log in to the system
Function flow	• User/Device will initiate a log in request. • The system will verify log in request using smart contracts • If the validation success, the system will allow the log in request.
Postcondition	Logged user or device can use system resources.

UC-3. Identification and monitoring health data

Use case element	Description
Title	Identification and Monitoring Health Data
Actors	Patient, Medical Device, Healthcare Provider
Goal	Collect health data from the registered medical device and patient for monitoring purposes.
Precondition	Patient and medical devices must have registration.
Trigger	Devices or healthcare providers will initiate to health data monitoring
Function flow	• Data will be collected from registered medical devices and patients. • The verified health data will monitor real time.
Postcondition	Patient health data will be monitored in real time.

UC-4. Transmit data to edge server

Use case element	Description
Title	Transmit health data to edge server
Actors	Medical device, Edge server
Goal	Transmit monitored health data from the medical device to the edge server for local processing and storage.
Precondition	Data from the medical device must be verified.
Trigger	Medical device completes data collection.

Function flow	• Medical device will generate data. • Data will be transmitted to edge server for process the data.
Postcondition	Health data will securely stored and processed by the edge server.

UC-5. Receive alert and notification

Use case element	Description
Title	Receive alert and notification
Actors	Patient, Healthcare Provider
Goal	The system will notify patients or healthcare providers about abnormal health data or critical alerts.
Precondition	Monitoring data must show abnormal results.
Trigger	Abnormal health data is detected.
Function flow	• The system will detect anomaly behavior in health data. • The system will generate an alert and send it to the patient and healthcare provider.
Postcondition	Patient and healthcare provider will notify of critical alerts.

UC-6. Access patient data

Use case element	Description
Title	Access patient data
Actors	Patient, healthcare provider, edge server
Goal	The system will only allow authorized actors to the patient data.
Precondition	All the initiated actors must be authenticated and authorized.
Trigger	Authorized actors request to patient data access
Function flow	• Patient or healthcare provider will request for patient data • Smart contract will validate access permissions. • Authorized actors will get patient data
Postcondition	Patient or healthcare provider will get patient data.

UC-7. Process patient data

Use case element	Description
Title	Process patient data
Actors	Edge server
Goal	Process patient health data for analytics, diagnosis, and treatment recommendations.
Precondition	Patient data must be transmitted and stored in the edge server.
Trigger	New data receives or processing request is made.
Function flow	• Edge server will process patient data. • Processed results will be stored securely and shared with authorized providers

Postcondition	Patient data will be stored in edge server.

UC-8. Request patient information

Use case element	Description
Title	Request patient information
Actors	Edge server, healthcare provider
Goal	Request specific patient information securely for diagnosis or treatment.
Precondition	The actor must be authenticated
Trigger	The actor requests specific patient information.
Function flow	• Healthcare provider will request for specific patient data. • The system will verify through smart contracts. • Requested data will be sent to the healthcare provider.
Postcondition	Healthcare providers will receive the requested patient information.

Component 2. MSCIDS (Multi-agent and Self-aware Collaborative Intrusion Detection Systems using Blockchain-enabled Semi-asynchronous Federated Learning) use case:

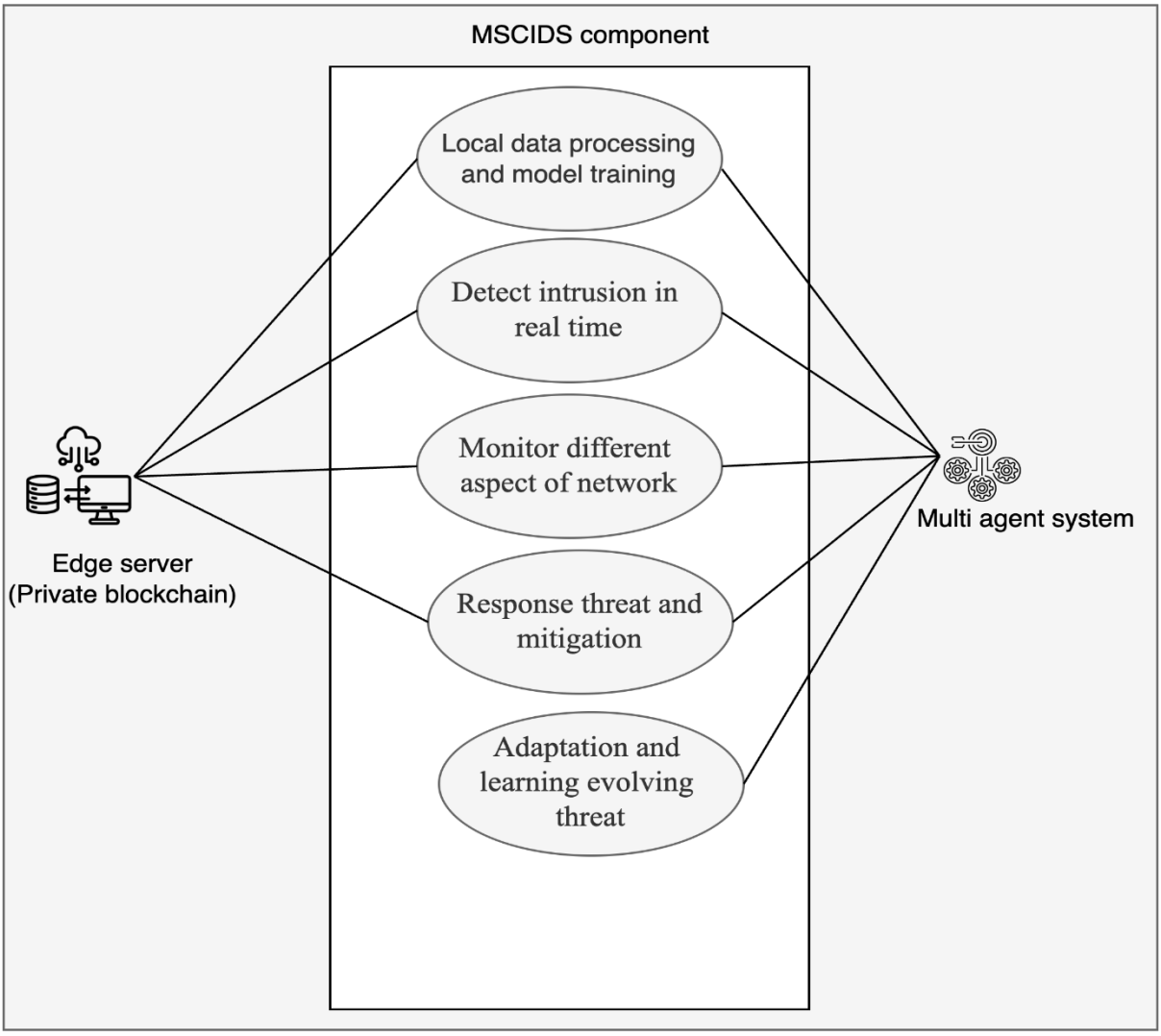

Figure 12: Use case diagram of MSCIDS component

UC-1. Local data processing and model training

Use case element	Description
Title	Local data processing and model training
Actors	Edge server, multi-agent system
Goal	Process local data at the edge server level and perform model training using FL algorithms to detect anomalies.
Precondition	Network data must be collected at the local edge server. Participating edge servers must be registered.

Trigger	New log data from component 1 and network data becomes available for processing.
Function flow	• Edge server will receive network data. • Data will be preprocessed by the local agent. • The model training process will be initiated using the received data. • Trained model parameters are updated locally.
Postcondition	Local models will be trained, and parameters will be updated for aggregation.

UC-2. Detect intrusion in real time

Use case element	Description
Title	Detect intrusion in real time
Actors	Multi-agent System, Edge Server
Goal	Analyze component 1 log data or network data in real time to detect intrusions or related threats using the trained FL model.
Precondition	Model must be trained and available for use at the edge server.
Trigger	Network activity that requires analysis.
Function flow	• Edge server will capture real-time network activity or component one log file. • Multi-agent system will analyze the captures data using the trained model. • The system will detect any abnormal activity or intrusions. • Alerts will be raised for detected threats
Postcondition	Intrusion will be either detected and logged or data will be marked as safe.

UC-3. Monitor different aspects of network

Use case element	Description
Title	Monitor different aspects of network
Actors	Edge server, Multi-agent system
Goal	Monitor different aspects of network traffic, including latency, data flow, and anomalies.
Precondition	Edge server must be operational and able to collect network data.
Trigger	Continuous network activity monitoring.
Function flow	• Edge server will collect real-time network traffic data. • Multi-agent system will monitor key network parameters like latency and throughput. • Abnormalities in network metrics will be flagged and analyzed.
Postcondition	Network traffic will be monitored and logged for further analysis.

UC-4. Response threat and mitigation

Use case element	Description

Title	Response threat and mitigation
Actors	Multi-agent System, Edge Server
Goal	Respond to detected threats in the network and perform appropriate mitigation measures based on predefined response strategies.
Precondition	Intrusion or threat must be detected by the system.
Trigger	Detection of an intrusion or anomaly in network traffic.
Function flow	• Intrusion will be detected by the system. • Multi-agent system will evaluate the threat and selects a response strategy. • Mitigation measures will be executed. • Threat details will be logged for audit and review.
Postcondition	The detected threat will be mitigated, and actions will be logged.

UC-5. Adaptations and learning of evolving threats

Use case element	Description
Title	Adaptations and learning of evolving threats
Actors	Multi-agent System, Edge Server
Goal	Adapt and improve the system to learn evolving threats using model updates.
Precondition	Threat data must be available and analyzed.
Trigger	New threats or anomalies are detected that require model adaptation.
Function flow	• Multi-agent system will identify new threat patterns. • Threat data will be collected and used to update local models. • Updated parameters will be shared through the FL process. • Global model will be updated using aggregated local updates.
Postcondition	The model will be updated to handle new threat patterns.

Component 3. DRMCS (Dynamic Risk Management and Communication and Sharing) use case:

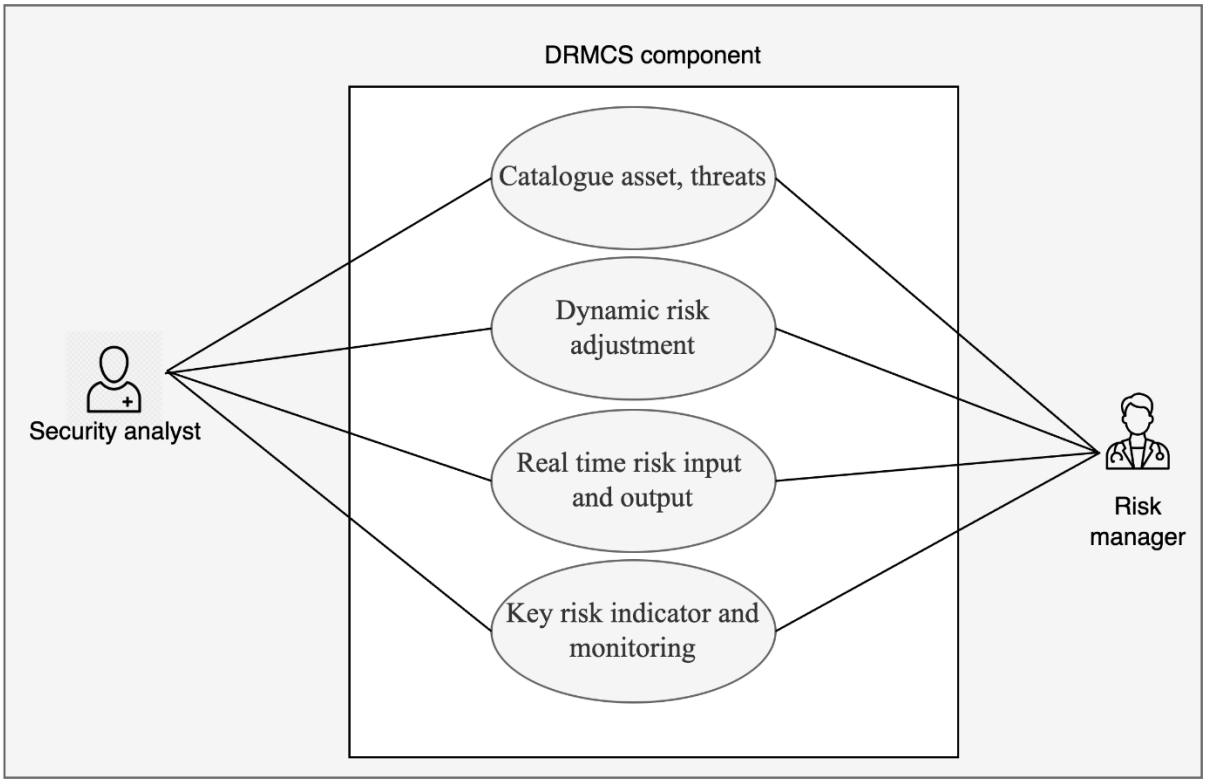

Figure 13: Use case diagram of DRMCS component

UC-1. Catalogue asset and threat

Use case element	Description
Title	Catalogue asset and threat information
Actors	Security analyst, Risk manager
Goal	The system will catalogue assets and potential threats for risk analysis and management.
Precondition	System components and data sources must be identified.
Trigger	New assets or threats are identified by the risk team.
Function flow	• Security analysts will identify new assets or threats. • Asset and threat information will be entered into the cataloging system. • The system will validate and update the asset/threat database.

	• Information will be shared with the FL network for global updates.
Postcondition	Asset and threat information will be catalogued and accessible for risk assessment.

UC-2. Dynamic risk adjustment

Use case element	Description
Title	Dynamic risk adjustment
Actors	Security analyst, Risk manager
Goal	The system will adjust the risk level dynamically based on evolving data and changing conditions.
Precondition	Risk assessment data and metrics must be available.
Trigger	Change in threat or asset status, or new data input from global model
Function flow	• The system will detect changes in asset or threat conditions. • Risk manager will initiate a dynamic risk assessment. • The system will use marisma framework models to adjust the risk level. • Adjusted risk level will be logged and shared with other systems for further monitoring.
Postcondition	Risk level will be adjusted to reflect the current situation.

UC-3. Real time risk input and output

Use case element	Description
Title	Real time risk input and output
Actors	Security analyst, Risk manager
Goal	The system will collect risk assessment information in real-time for immediate response.
Precondition	The system must be actively monitoring risk inputs and have the ability to calculate risk outputs.
Trigger	Continuous monitoring or new threat input.
Function flow	• The System will collect real-time risk data. • The system will generate risk output and alerts the Security Analyst or Risk Manager. • Risk mitigation suggestions will be provided in real-time.
Postcondition	Real-time risk status will be updated, and appropriate alerts will be issued.

UC-4. Key risk indicator and monitoring

Use case element	Description
Title	Key risk indicator and monitoring
Actors	Security analyst, Risk manager

Goal	Define and monitor key risk indicators to assess the risk profile and determine trends in risk evolution.
Precondition	Key risk indicator must be defined and metrics must be measurable.
Trigger	Specific changes in monitored variables.
Function flow	• Risk Manager will define key risk indicators for monitoring. • The system will collect data to calculate key risk indicator metrics. • The system will analyze the key risk indicator and visualizes the trend in risk. • The system will alert if key risk indicator thresholds are exceeded.
Postcondition	Key risk indicators will be continuously monitored, and alerts will be generated when necessary.

4.2 Dependencies and Interaction among key Components

In this subsection, we have discussed the key components of Di4SPDS and their interaction.

4.2.1 Component 1: BSCDA (Blockchain and Smart-contract enabled Cross-Domain Authentication and Access Control Scheme):

Input type: Log File

Explanation: The blockchain is deployed and running on the docker that is connected with Kubernetes through API. The Kubernetes helps to deploy the blockchain nodes in the network and provides flawless communication between the connected nodes on the blockchain network. In the blockchain, when a node attempts to access the network or the content stored on the blockchain storage, a log is generated for the node and records all attempted requests. In the blockchain, only authorized nodes can access the network and files stored in it by default. However, some malicious actors on the blockchain try to access unauthorized content/block or modify the blockchain properties. The log file can be analyzed to identify those malicious actors on the blockchain. These logs, being immutable and cryptographically secured, provide a detailed account of who attempted to access what, when, and whether it was allowed. The data from the log file, with its comprehensive and detailed nature, can be used to determine attempt of unauthorized access by malicious users or device with the user ID or device ID. The following attributes of the log will be processed in the IDS to identify and categorize malicious activity.

- **Used_Id:**
  - The identity of the user or device attempting access.
  - **Purpose:** to Identify the registered user.

- **Time_stamp:**
  - Date and time of the access attempt.
  - **Purpose:** Tracks the timing of events and detects anomalies related to specific time frames, such as repeated failed attempts in a short duration**.**
- **Access_level:**
  - The type of the access permission requested by the user such as read, write or update.
  - **Purpose:** Verify the access level of the user.
- **Resource_access:**
  - The resource, system, or data requested by the user.
  - **Purpose:** To track the alteration on the content created by the user on the content, file , or system.
- **Authentication_status:**
  - Approval or disapproval of the request to access the request.
  - **Purpose:** To verify the authentication is Approved or disapproved.
- **Authentication_perpose:**
  - The reason for the requesting the access on the system.
  - **Purpose**: To track the purpose for access request by the user and what it actually requesting and doing with content.
- **Reconnection_attempts:**
  - **F**requency of the reconnection attempts.
  - **Purpose:** Detect abnormal numbers of reconnections which may indicate a DDoS attack or a network intrusion attempting to cause disruption
- **Error_and _Warning_Messages:**
  - Count and type of error or warning events
  - **Purpose:** Detect frequent failures, such as TLS handshake failures, certificate verification issues, and context deadline exceeded errors. A high rate of these events may indicate a possible attack or misconfiguration.
- **TLS_Handshake_Failures**
  - Frequency of TLS handshake failures
  - **Purpose:** A high number of TLS handshake failures can signal a potential attack such as MITM (Man-in-the-Middle) or unauthorized access attempts.
- **Context_Deadline_Exceeded**
  - Count of context deadline exceeded events
  - **Purpose:** This might indicate network delays, Denial of Service (DoS), or attempts to overwhelm the network by delaying communication.
- **Failed_Proposal_Validation**
  - Count of failed transaction proposals

  - **Purpose:** Failed transaction proposals may indicate attempts to submit malicious transactions or abuse system behavior
- **Failed_Certificate_Verification**
  - Count of failed certificate verifications
  - **Purpose:** Repeated failures in verifying certificates might indicate identity spoofing attempts or improperly configured peers
- **Transaction_Endorsement_and_Response_Time**
  - Duration of Smart contract endorsement and responses
  - **Purpose:** Abnormal delays or quick responses during smart contract processing may indicate tampering or suspicious activity
- **Invalid Identity Warnings**
  - Occurrences of invalid identities
  - Purpose: Helps in detecting spoofed identities, unauthorized access attempts, or certificate tampering.

Following is an example of a log file:

2024-10-01 09:22:18.879 UTC INFO [auth] -> User1 from Org1MSP is attempting to access the ledger 2024-10-01 09:22:18.879 UTC INFO [access] -> User1 requested read access on ledger_data_block45 2024-10-01 09:22:18.880 UTC INFO [auth] -> Access approved for User1 from Org1MSP 2024-10-01 09:22:18.881 UTC INFO [chaincode] -> User1 invoked chaincode 'basic' on channel 'mychannel' with transaction ID abc123 2024-10-01 09:22:18.900 UTC INFO [endorser] -> Endorsement success for transaction ID abc123 by User1 from Org1MSP 2024-10-01 09:23:19.103 UTC WARN [auth] -> Reconnection attempt detected for User1, attempt 1 2024-10-01 09:23:19.104 UTC WARN [auth] -> Reconnection attempt detected for User1, attempt 2 2024-10-01 09:23:19.150 UTC INFO [auth] -> User2 from Org2MSP is attempting write access on ledger_data_block46 2024-10-01 09:23:19.151 UTC ERROR [auth] -> Write access disapproved for User2 from Org2MSP 2024-10-01 09:23:19.152 UTC WARN [auth] -> Failed certificate validation for User2 from Org2MSP 2024-10-01 09:25:19.451 UTC INFO [tls] -> TLS handshake success for User1 from Org1MSP on peer0.org1.example.com 2024-10-01 09:25:19.500 UTC WARN [chaincode] -> Invalid identity warning for device1.org2.example.com during chaincode update request 2024-10-01 09:25:19.550 UTC INFO [endorser] -> Endorsement success for transaction ID def456 by device1.org2.example.com

For the analysis of the above-mentioned attributes from the log file, first, it will be collected from the docker or Kubernetes where the blockchain framework is installed. Then, it will be sent to the parsing tool, which produces a CSV/JSON file that will be used in component 2.

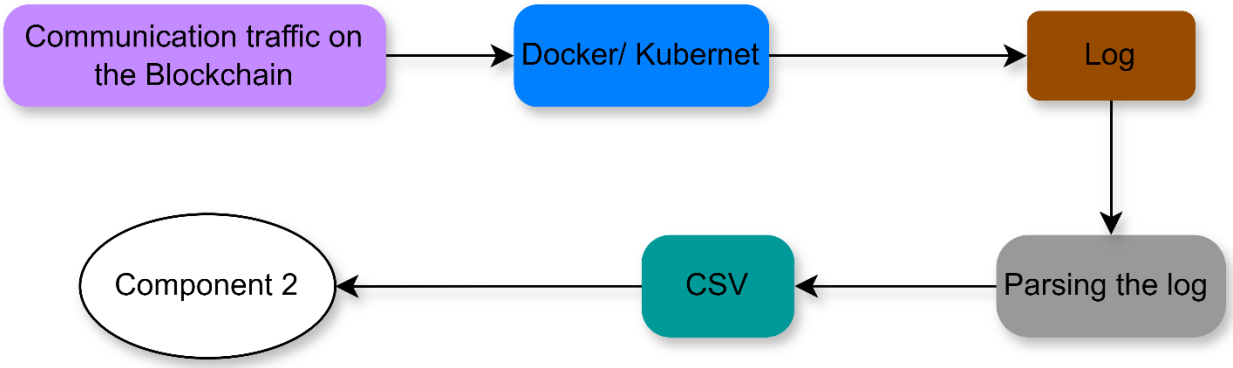

Figure 14: Process of Component 1 for input and output elements

4.2.3 Component 2: MSCIDS (Multi-agent and Self-aware Collaborative Intrusion Detection Systems using Blockchain-enabled Semi-asynchronous Federated Learning):

Input: Blockchain logs data from component 1 and Real-time Network Data (Packets)

Blockchain logs data from component 1:

Step 1:

- Component 1 will generate JSON logs of relevant events, such as authentication attempts, access control violations, and other events.
- These logs will be sent to Component 2 via an API call. In our project, we will use REST API, so the API looks like a POST request where Component 1 sends the JSON logs to Component 2.

Step 2:

- Component 2 will receive the JSON logs via the API and handle the data by converting it from JSON to CSV format.
- **Example Python script**:
  - The Python script will read the JSON logs, extract relevant fields, and convert them into a structured CSV format.
  - This CSV file will later be merged with the PCAP data for use as the final dataset for running the IDS.

Real-time Network Data:

Packet Data (PCAP Files): This data contains detailed information about individual packets transmitted over the network.

Format: **PCAP** (Packet Capture Format) stores packet-level data.

Tools: Wireshark, and **Argos** generate PCAP files.

Data fields are:

- **Timestamp**: When the packet was captured.
- **Source IP Address**: IP address of the sender.
- **Destination IP Address**: IP address of the receiver.
- **Protocol**: The protocol used (e.g., TCP, UDP, ICMP).
- **Source Port**: Port on the source device.
- **Destination Port**: Port on the destination device.
- **Packet Size**: Size of the packet (in bytes).
- **Payload**: The actual content/data being transmitted.

The Python script will read the above PCAP files, extract relevant fields, and convert them into a structured CSV format.

Finally, the PCAP data and Blockchain logs in CSV format merge the two datasets to create a combined CSV file as input data for running the IDS.

Output: Detected threats and vulnerabilities

Format: Json format

Fields are:

- **threat_id**: Unique identifier for the detected threat.
- **user_id**: ID of the user or entity affected.
- **device_id**: Device where the threat was detected.
- **detected_at**: Timestamp when the threat was detected.
- **threat_type**: Type of threat (e.g., data poison, DDoS).
- **Threat_description**: A brief description about threat
- **severity**: Severity level (e.g., low, medium, high, critical).
- **actions_taken**: What actions were taken (e.g., access revoked, device restricted).
- **status**: Whether the threat was mitigated, or ongoing.

Fileds in Json format:

```
1. {
2.   "threat_id": "T123456",
```

```
3.  "user_id": "user123",
4.  "device_id": "deviceABC",
5.  "detected_at": "2024-10-14T12:37:00Z",
6.  "threat_type": "DDoS",
7.  “threat_description: ” Description of the DDos“,
8.  "severity": "high",
9.  "actions_taken": "Access revoked, device quarantined",
10.  "status": "mitigated"
11. }
```

The above Json file will pass to Component 3 through API call for further processing.

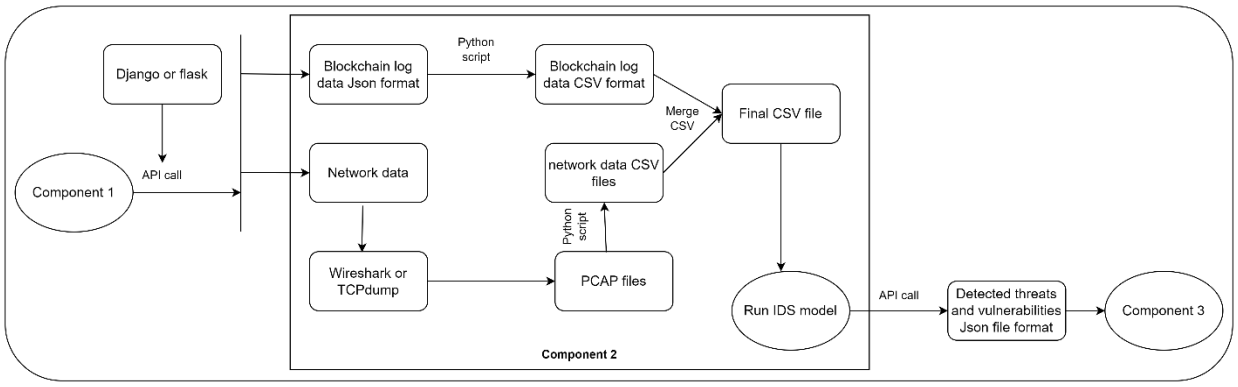

Figure 15 : Input output data format for component 2

4.2.4 Component 3 – DRMCS (Dynamic Risk Management and Communication and Sharing)

The component will have two types of possible inputs for which the necessary communication APIs will be implemented:

1. Security events: The component will receive the Json file of threats and vulnerabilities reported by Component 2. Based on each reported threat and vulnerability, the risk level of the system shall be automatically and dynamically recalculated. The output of the system will be an alert indicating that a variation in the risk level has occurred, which can be propagated to the different nodes of the system.

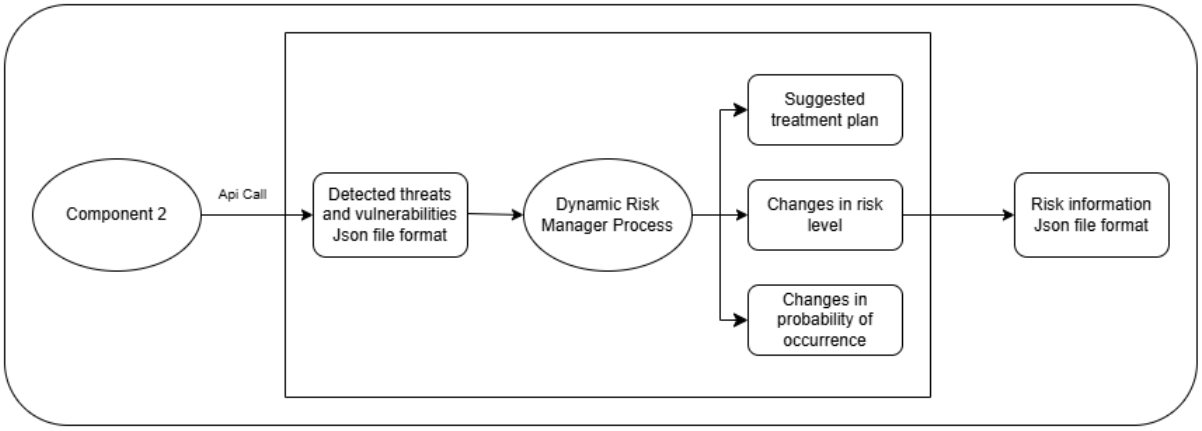

Figure 16: Input output data format for component 3

Administrators will be able to access the tool to consult in detail the variations produced through a dashboard, and also a JSON file will be generated. The structure of the output file is as follows

JSON structure output after a security threat reported, which includes:

- A **treatment plan** with multiple security controls and their priority order.
- A **risk level** with two fields: previous risk level and current risk level, both ranging between 0 and 100.
- The **change in probability of occurrence** for multiple threats, each having an initial value and a final value.

Explanation:

(1) **Incident Response** (incident_response): The top-level object that encapsulates the response after the security incident.
(2) **Treatment Plan** (treatment_plan):
    a. **Controls** (controls): A list of security controls that need to be applied, each having:
        i. **control_id**: A unique identifier for the control.
        ii. **control_name**: The name of the control.
        iii. **description**: A brief explanation of the control.
        iv. **priority_order**: The order of priority for implementing the control (1 = highest priority).
(3) **Risk Level** (risk_level):
    b. **previous_risk_level**: The level of risk before applying the controls, on a scale of 0 to 100.

c. **current_risk_level**: The level of risk after applying the controls, on a scale of 0 to 100.

(4) **Probability Change** (probability_change): A list of threats where the probability of occurrence has changed. Each entry contains:

d. **threat**: The name of the threat.

e. **initial_value**: The probability of occurrence before the treatment (between 0 and 100).

f. **final_value**: The probability of occurrence after the treatment (between 0 and 100).

Flexibility:

- You can add more controls to the treatment plan, adjusting the **priority_order** as needed.
- The **risk level** is designed to reflect how effectively the treatment plan reduces the overall risk, based on a 0-100 scale.
- The **probability change** section allows you to track how the probability of specific threats has shifted due to the implemented controls.

This structure can be adapted or extended depending on the complexity of your cybersecurity incident response process.

```
{
  "incident_response": {
    "treatment_plan": {
      "controls": [
        {
          "control_id": "C001",
          "control_name": "Patch Management",
          "description": "Apply security patches to mitigate vulnerabilities.",
          "priority_order": 1
        },
        {
          "control_id": "C002",
          "control_name": "Firewall Configuration",
          "description": "Update firewall rules to prevent unauthorized access.",
          "priority_order": 2
        },
        {
          "control_id": "C003",
          "control_name": "Employee Training",
```

```
      "description": "Conduct security awareness training for employees.",
      "priority_order": 3
    }
  ]
 },
 "risk_level": {
  "previous_risk_level": 85,
  "current_risk_level": 60
 },
 "probability_change": [
  {
   "threat": "Phishing Attack",
   "initial_value": 80,
   "final_value": 50
  },
  {
   "threat": "SQL Injection",
   "initial_value": 70,
   "final_value": 40
  },
  {
   "threat": "Ransomware",
   "initial_value": 90,
   "final_value": 70
  }
 ]
}
}
```


2. CVE recordsets: Mechanisms will be in place to read vulnerability logs and filter out vulnerabilities that may affect DDS systems. Based on these vulnerabilities, the system will generate alerts and security recommendations as a preventive mechanism.

Here's the **JSON structure** as input when a new **CVE** appears. The structure includes:

(1) The **threat** associated with the CVE.
(2) **Assets** that may be affected by the CVE.
(3) For each asset, the **impact values** for the dimensions of **confidentiality, integrity, and availability** on a scale of high, medium, or low.
(4) **Impact metrics** related to the CVE.

Explanation:

(1) **CVE Analysis Input** (cve_analysis_input): The top-level object that contains all information related to the new CVE.
(2) **CVE ID** (cve_id): The identifier for the specific CVE, following the format **CVE-[Year]-[ID]**.
(3) **Threat** (threat):
    a. **threat_name**: The name or type of threat that the CVE represents (e.g., "Remote Code Execution").
    b. **description**: A brief description of the nature of the threat and how it might be exploited.
(4) **Affected Assets** (affected_assets): A list of assets that might be impacted by this CVE. Each asset includes:
    c. **asset_id**: A unique identifier for the asset.
    d. **asset_name**: The name of the asset potentially affected.
    e. **impact_dimensions**: The impact on the asset for each of the following dimensions:
        i. **confidentiality**: Impact level (high, medium, or low).
        ii. **integrity**: Impact level (high, medium, or low).
        iii. **availability**: Impact level (high, medium, or low).
(5) **CVE Impact Metrics** (cve_impact_metrics):
    f. **cvss_score**: The overall CVSS (Common Vulnerability Scoring System) score, indicating the severity of the vulnerability (e.g., 9.8).
    g. **exploitability_score**: The exploitability score, showing how easily the vulnerability can be exploited.
    h. **impact_score**: The impact score, indicating the potential impact on the system.
    i. **vector_string**: The CVSS vector string, representing the specific attributes of the vulnerability (e.g., AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H).
    j. **description**: A brief explanation of the CVE's metrics and its overall criticality.

Flexibility:

- The **impact_dimensions** for each asset provide a clear understanding of how confidentiality, integrity, and availability are affected on a scale of high, medium, or low.
- The **cve_impact_metrics** section provides additional context through **CVSS scores** and an explanation of the vulnerability's severity.

This structure is flexible and can accommodate additional fields or metrics if needed, depending on the specific requirements of your system.

```
{
  "cve_analysis_input": {
    "cve_id": "CVE-2024-12345",
    "threat": {
      "threat_name": "Remote Code Execution via CVE-2024-12345",
      "description": "This CVE allows an attacker to remotely execute arbitrary code in the affected software."
    },
    "affected_assets": [
      {
        "asset_id": "A001",
        "asset_name": "Web Application Server",
        "impact_dimensions": {
          "confidentiality": "high",
          "integrity": "high",
          "availability": "medium"
        }
      },
      {
        "asset_id": "A002",
        "asset_name": "Customer Database",
        "impact_dimensions": {
          "confidentiality": "high",
          "integrity": "medium",
          "availability": "low"
        }
      }
    ],
    "cve_impact_metrics": {
      "cvss_score": 9.8,
      "exploitability_score": 8.6,
      "impact_score": 9.2,
      "vector_string": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H",
      "description": "This CVE has a high impact due to the critical nature of remote code execution with no privileges required."
    }
  }
```

```
36. }
37.
```

Possible outputs:

Output type 2.1:

Patch Availability and Mitigation Guidance

- **Value**: Knowing whether patches or workarounds exist for a CVE can help organizations quickly implement fixes.
- **How to Implement**: Leverage the **NVD (National Vulnerability Database)** or vendor advisories (Microsoft, Oracle, etc.) to check for available patches or mitigation measures.
- **Example**:

```
1. "patch_availability": {
2.   "patch_available": true,
3.   "patch_release_date": "2024-10-01",
4.   "patch_link": "https://vendor.com/security-update",
5.   "alternative_mitigation": "Disable vulnerable service if patching is not possible."
6. }
7.
```

Output type 2.2:

Asset-Specific Criticality

- **Value**: CVE impact may vary depending on the **criticality of the asset** it affects. Enhancing your system to account for the role of an asset in the organization (e.g., high-value assets) can provide better prioritization.
- **How to Implement**: Assign **asset criticality ratings** (e.g., business-critical, high, medium, low) and calculate the overall risk score accordingly.
- **Example**:

```
1. "asset_criticality": {
2.   "asset_id": "A001",
3.   "asset_name": "Financial Transaction Server",
4.   "criticality": "critical",
5.   "adjusted_risk_score": 95
6. }
7.
```

Output type 2.3:

Automated Risk Scoring and Prioritization

- **Value**: You can automatically calculate a **risk score** for each CVE based on factors like exploitability, asset criticality, and business impact. This helps prioritize which vulnerabilities to address first.
- **How to Implement**: Develop an algorithm or scoring model that takes into account CVSS scores, exploit availability, asset criticality, threat intelligence, and vulnerability chaining.

- **Example**:

```
1. "risk_score": {
2.   "cvss_base_score": 9.8,
3.   "exploitability_score": 8.6,
4.   "asset_criticality_adjustment": 10,
5.   "overall_risk_score": 95
6. }
```

Component	Data type	Data format	Fields
Component 1: BSCDA	Input	Network packets	All the activities on the network by user, device, and system
	Output	Log file -> Json	• User Id • Timestamp • Access_level • Resource_access • Authentication_statuts • Authentication_purpose • Reconnection_attempts • Error_and_warning_message • TLS_handshake_failures • Context_deadline_exceeded • Failed_proposal_validation • Failed_certificate_validation • Failed_certificate_verification • Transaction_endorsement_and_response • Invalid_identity_warning
	Input	Json	• Blockchain logs data from component 1
			• Timestamp • Source_IP_address • Destination_IP_address • Protocol

Component 2: MSCIDS		PCAP files->Json	• Source_port • Destination_port • Packet_size • Payload
	Output	Json	• Threat_id • User_id • Device_id • Detected_at • Threat_type • Threat_description • Severity • Action_taken • Status
Component 3 – DRMCS	Input	Json	• Threat_id • Threat_name • Description • Asset_id • Asset_name • Asset_description • Dimenstion_name • impact
	Output	Json	• Incident_response • Treatment_plan • Control_id • Control_name • Description • Priority_order • Risk_level • Previous_risk_level • Current_risk_level • Probability_change • Threat • Intiaial_value • Final_value

Table: Input-output data type and data format for every component

5. Conclusion

The Di4SPDS project deliverable D2.2 represents a comprehensive effort to enhance security, privacy, and efficiency in decentralized and distributed systems (DDS) through innovative, integrated solutions. By leveraging blockchain technology, multi-agent intrusion detection systems, and dynamic risk management, this framework addresses the critical challenges of cross-domain access control, proactive threat detection, and real-time risk assessment. Each component—blockchain-based access control, federated learning-enabled collaborative intrusion detection, and a dynamic risk management system—interacts seamlessly to ensure robust security and adaptability across diverse domains, including healthcare, finance, and industrial sectors. Together, these methods position Di4SPDS as a model for safeguarding DDS environments, setting a benchmark for sustainable cybersecurity in distributed infrastructures and creating a foundation for further advancements in secure, transparent, and resilient system architectures.

References

[FU1] Sumithra V, Shashidhara R, Mukhopadhyay D (2022) Design of a secure and privacy preserving authentication protocol for telecare medical information systems. Secur Privacy 5(4):228.

[FU2] Indushree M, Raj M, Mishra VK, Shashidhara R, Das AK, Bhat V (2022) Mobile-chain: secure blockchain based decentralized authentication system for global roaming in mobility networks. Comput Commun 200:1.

[FU3] Amin R, Islam SH, Gope P, Choo K-KR, Tapas N (2018) Anonymity preserving and lightweight multimedical server authentication protocol for telecare medical information system. IEEE J Biomed Health Inform 23(4):1749–1759.

[FU4] Chen Y, Ding S, Xu Z, Zheng H, Yang S (2019) Blockchain-based medical records secure storage and medical service framework. J Med Syst 43(1):1–9.

[FU5] Giri D, Maitra T, Amin R, Srivastava P (2015) An efficient and robust rsa-based remote user authentication for telecare medical information systems. J Med Syst 39(1):1–9.

[FU6] Tan Z (2014) A user anonymity preserving three-factor authentication scheme for telecare medicine information systems. J Med Syst 38(3):1–9.

[FU7] Yoon E-J, Yoo K-Y (2013) Robust biometrics-based multi-server authentication with key agreement scheme for smart cards on elliptic curve cryptosystem. Supercomput 63(1):235–255.

[FU8] Fan C-I, Lin Y-H (2009) Provably secure remote truly three-factor authentication scheme with privacy protection on biometrics. IEEE Trans Inf For Secur 4(4):933–945.

[FU9] Lin C, He D, Huang X, Choo K-KR, Vasilakos AV (2018) Bsein: a blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. J Netw Comput Appl 116:42–52.

[FU10] Kuo T-T, Kim H-E, Ohno-Machado L (2017) Blockchain distributed ledger technologies for biomedical and health care applications. J Am Med Inform Assoc 24(6):1211–1220.

[FU11] Zhang A, Lin X (2018) Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain. J Med Syst 42(8):1–18.

[FU12] Fan K, Wang S, Ren Y, Li H, Yang Y (2018) Medblock: efficient and secure medical data sharing via blockchain. J Med Syst 42(8):1–11.

[UCLM13] Klaus Pohl. 2010. Requirements Engineering: Fundamentals, Principles, and Techniques. Springer. ISBN: 3662518880, 978-3662518885

[UCLM14] Karl Wiegers. 2003. Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle. Microsoft Press. ISBN 0735618798, 9780735618794

[UCLM15] D.G. Rosado, L.E. Sánchez, Á.J. Varela-Vaca, A. Santos-Olmo, M.T. Gómez-López, R.M. Gasca, E. Fernández-Medina, Enabling security risk assessment andmanagement for business process models, J. Inf. Secur. Appl. 84 (2024) 103829,http://dx.doi.org/10.1016/j.jisa.2024.103829

[UCLM16] D.G. Rosado, A. Santos-Olmo, L.E. Sánchez, M.A. Serrano, C. Blanco, H.Mouratidis, E. Fernández-Medina, Managing cybersecurity risks of cyber-physicalsystems: The MARISMA-CPS pattern, Comput. Ind. 142 (2022) 103715,http://dx.doi.org/10.1016/j.compind.2022.103715

[UCLM17] D.G. Rosado, J. Moreno, L.E. Sánchez, A. Santos-Olmo, M.A. Serrano, E.Fernández-Medina, MARISMA-BiDa pattern: Integrated risk analysis for bigdata, Comput. Secur. 102 (2021) 102155,http://dx.doi.org/10.1016/j.cose.2020.102155

[UCLM18] J. Overbeek, Meta Object Facility (MOF): Investigation of the State of the Art,University of Twente, 2006

[UCLM19] ISO/IEC 27001:2013, SO/IEC 27001:2013 - Information Technology — SecurityTechniques — Information Security Management Systems — Requirements,Standard, International Organization for Standardization, Geneva, CH, 2013,Iso/Iec 2013.

[UCLM20] Magerit, Magerit_v3: Methodology for information systems risk analysis andmanagement. The method, in: Magerit_V3, Ministry of Public Administration,2012, URLhttp://administracionelectronica.gob.es/.

[UCLM21] R. Bendraou, B. Combemale, X. Cregut, M.-P. Gervais, Definition of an executableSPEM 2.0, in: 14th Asia-Pacific Software Engineering Conference, APSEC'07,2007, pp. 390–397,http://dx.doi.org/10.1109/ASPEC.2007.60

[UCLM22] M. Ross, A.J. Jara, A. Cosenza, Baseline Security Recommendations for IoT inthe Context of Critical Information Infrastructures, (November) European UnionAgency For Network And Information Security, 2017,http://dx.doi.org/10.2824/03228