# D5.3 Trustworthy data lakes federation

## Third Release Report

Revision: v1.0

| | |
|---|---|
| **Work package** | WP 5 |
| **Task** | Task 5.1, 5.2, 5.3 |
| **Due date** | 31/05/2025 |
| **Submission date** | 31/05/2025 |
| **Deliverable lead** | TUB |
| **Version** | 0.8 |
| **Authors** | Fernando Castillo (TUB), Fabian Piper (TUB), Sepideh Masoudi (TUB): all partners and pilot cases |
| **Reviewers** | CYBERNETICA, CEFRIEL |
| **Abstract** | This deliverable covers the final report and evaluation of trustworthy data lakes federations on the TEADAL context. |
| **Keywords** | Trustworthiness, Evidence Collection, Privacy Preserving Computations. |

WWW.TEADAL.EU

Grant Agreement No.: 101070186
Call: HORIZON-CL4-2021-DATA-01

Topic: HORIZON-CL4-2021-DATA-01-01
Type of action: HORIZON-RIA

## Document Revision History

| Version | Date | Description of change | List of contributor(s) |
|---|---|---|---|
| V0.1 | 16/12/2024 | Document initial structure and ToC. | TUB |
| V0.5 | 22/04/2025 | First draft with initial content. | All partners |
| V0.8 | 15/05/2025 | First version for internal review. | TUB, All partners |
| V1.0 | 29/05/2025 | Addressing of internal reviewers comments | TUB, All partners |

## DISCLAIMER

Funded by the European Union

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,**
**Research and Innovation SERI**

Funded by the European Union (TEADAL, 101070186). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. This work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI).

## COPYRIGHT NOTICE

| **Project funded by the European Commission in the Horizon Europe Programme** | | |
|---|---|---|
| **Nature of the deliverable:** | **R** | |
| **Dissemination Level** | | |
| **PU** | *Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)* | ✔ |
| **SEN** | *Sensitive, limited under the conditions of the Grant Agreement* | |
| **Classified R-UE/ EU-R** | *EU RESTRICTED under the Commission Decision No2015/ 444* | |
| **Classified C-UE/ EU-C** | *EU CONFIDENTIAL under the Commission Decision No2015/ 444* | |
| **Classified S-UE/ EU-S** | *EU SECRET under the Commission Decision No2015/ 444* | |

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

DATA: Data sets, microdata, etc.

DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

OTHER: Software, technical diagram, algorithms, models, etc.

# EXECUTIVE SUMMARY

This document presents the third and final iteration of the TEADAL Trust Plane, an evidence-based framework that lets organisations exchange data across federated lakes, edge to cloud, while guaranteeing verifiable identity, confidentiality, provenance and energy awareness. Building on the foundations laid in the first two releases, D5.3 consolidates the trust-enabling mechanism into a three-layer model: identity, computational and behavioural trust.

Key enhancements in this iteration include a blueprint for deploying single TEADAL nodes and scaling them into federations, an implementation of TrustOps workflow that now span development, attestation and runtime monitoring, and an expanded evidence model that links actions on data, software or infrastructure to verifiable proofs. These improvements give Data-Lake Operators a configurable menu of trust controls, allowing each federation to balance cost, audit-data volume and verification depth without losing interoperability.

D5.3 also reports validation for specific scenarios based on pilot use cases and independent from the use cases scenarios, showing that the final Trust Plane meets the requirements defined in earlier deliverables. Measured latencies, gas costs and scalability benchmarks confirm that evidence production, anchoring and verification remain economical and performant at real-world scale.

In summary, the third iteration demonstrates the TEADAL Trust Plane as a promising system prototype for trustworthy and energy-aware data sharing across decentralized environments.

Funded by the European Union

## Table of Contents

Funded by the European Union

## LIST OF FIGURES

Funded by the European Union

Funded by the European Union

# ABBREVIATIONS

| | |
|---|---|
| **BPMN** | Business Process Model and Notation |
| **FDP** | Federated Data Product |
| **sFDP** | shared Federated Data Product |
| **ML** | Machine Learning |
| **DLO** | Data Lake Operator |
| **DLT** | Distributed Ledger Technology |
| **TNS** | TEADAL Name Service |
| **EVM** | Ethereum Virtual Machine |
| **GDPR** | General Data Protection Regulation |
| **MPC** | Multi-Party Computation |
| **zk-SNARK** | Zero Knowledge Succinct Non-Interactive Arguments of Knowledge |
| **TEE** | Trusted Execution Environment |
| **IPFS** | InterPlanetary File System |
| **DSWE** | Data Synthesis Workflow Engine |
| **RSD** | Realistic Synthetic Data |
| **CSD** | Causal Synthetic Data |
| **ASD** | Artificial Synthetic Data |
| **HSD** | Hybrid Synthetic Data |
| **DID** | Decentralized Identifier |
| **WP** | Work Package |
| **ZKP** | Zero-Knowledge Proof |

Funded by the European Union

# 1 Introduction

Data lakes collect a vast amount of data and are characterized by (i) potential distribution across various storage systems, (ii) diverse data formats (structured, semi-structured, unstructured), (iii) possibly absent or inconsistently formatted metadata, and (iv) typically dynamic and autonomously changing data [3]. They emerged as an approach to handle data originating from diverse sources, such as social networks, edge devices or enterprise applications. TEADAL ensures trusted, mediationless, verifiable, and energy-efficient data exchange in standalone as well as federated data lakes, particularly those deployed across the computing continuum. Its architecture builds around the concept of data mesh, augmented with stretchable and federated data products. To this end, established tools and data products ensure a seamless integration of TEADAL tools across heterogeneous environments.

Trust can be defined in many ways. In the context of software systems, it describes a complex interplay of beliefs, expectations, and reliability of systems, in particular the reliance on a software's ability to meet specified requirements, even amid uncertainty. In TEADAL and in the context of this deliverable, we characterize trust through three foundational layers: identity trust, which confirms the authenticity of entities; computational trust, which guarantees the integrity and confidentiality of computations; and behavioral trust, which assesses trustworthiness based on past actions and observed behavior. The data exchange process typically involves multiple parties. In this context, trust must be enabled across different organizations, across multiple networks or different device boundaries. The involved parties need trust that a data exchange is performed according to predefined terms. In distributed and decentralized data exchange procedures, applying federated data products on top of a data mesh architecture alone is not enough to ensure trustworthiness.

In TEADAL, the Trust Plane has been proposed to allow all parties involved in a data exchange to verify that a piece of software that has performed a particular operation behaves as expected. Evidence serves as a fundamental pillar for trust within TEADAL, emphasizing its need for variety, veracity, and ubiquitous collection. Moreover, the authenticity of the collected evidence is strengthened by ensuring authentication and integrity in the evidence. While verifiability of evidence through verifiable credentials facilitates trust in certain claims, authorization governs access rights and distribution enhances security and reliability guarantees. A key advancement in this regard is the generation and collection of verifiable evidence throughout the TEADAL infrastructure.

This document reports on the third iteration of the TEADAL evidence-based trust framework for federated data exchange known as the TEADAL Trust Plane. Building on the work of the first and second iteration [1, 2], in the third iteration we focussed on validating and demonstrating the TEADAL Trust Plane when having a lack of trust, which hinders data exchange in distributed, decentralised environments such as federated data lakes. We have further investigated, developed and finalized the mechanisms necessary for parties wanting to exchange data to gain confidence that data are stored, handled and exchanged. In the following we provide an overview of these mechanisms and how they fit into the Trust Plane architecture, whereas the remainder of the document delves into the details. Based on these results, we performed a cost and energy evaluation of different designs of the trustworthy infrastructure to guide adopters of the TEADAL technology. Moreover, we validated the trustworthy infrastructure in use case dependent and use case independent scenarios.

Funded by the European Union

# 2 Overview

In the first iteration of TEADAL, we introduced trust models and evidence that we are collecting to enable all stakeholders to gain trust in the correct exchange of data. We focused on the collection of runtime evidence, generated through the execution of FDPs and the storage and distribution of this evidence, e.g., either anchored in a private blockchain or on a public one. In the second iteration, we focused on extending this evidence, by showcasing how we can authenticate different stakeholders through decentralized identifies and how this evidence can be aggregated and verified continuously by utilizing privacy-preserving infrastructure, such as trusted execution environments (TEEs), secure multi-party computation (MPC), and zero-knowledge proofs (ZKP). In this third and last iteration of TEADAL, we conducted a final refinement and evolution of our evidence-based trustworthy architecture, that for example takes into account federation-specific interactions. Towards this end, we describe in this section the general requirements that we want to achieve for the Teadal Trust Plane in terms of trustworthiness, derived from the pilot cases specified in D2.3 [7].

## A. General Trustworthiness Requirements

### i. Track and monitor data usage

Trust requires comprehensive tracking and monitoring of data usage. This is achieved by recording evidence of data access, processing, and manipulation. Therefore, auditable records are needed that show how data is stored, accessed, and used, enabling federation members and third parties to review these activities. The record is essential for preventing unauthorized actions and establishing accountability. To ensure transparency of data access across various system levels and allow entities to understand the flow and usage of data, it should be accompanied by a mechanism that allows to determine who accessed or requested data. Additionally, data management requires definition and enforcement of policies. Tracking and monitoring ensures that data access complies to established policies, by facilitating the verification of adherence. From a data owner's perspective, an FDP should provide an interface for managing data sharing consent policies. From an external user's perspective, there must be a way to access data they have been granted.

### ii. Ensure privacy and confidentiality

To eliminate reliance on subjective or unverifiable trust, cryptographic means for technical realization ensure that data-sharing and processing comply with privacy guarantees. Therefore, a variety of computation methods, including privacy-preserving mechanisms, should be provided for different datasets. Furthermore, technical components, such as tools for anonymization, anonymity checks, and consent verification should facilitate privacy-compliant data access. To enforce data confidentiality according to access rules, the solution must implement policies, thereby preventing unauthorized access to raw data. It should define the level of confidentiality of data, KPIs, and reports. Access to confidential data must be prohibited for unauthorized users. Data aggregation must have a minimum threshold to prevent analysis on small datasets. Additionally, each entity should be able to restrict data sharing with other entities. To comply with privacy regulations, it is required to only allow sharing of data based on provided consent, verifying consent information and its relation to sensitive information. This should also support compliance with privacy regulations by representing consent for access control and implementing policies based on data to enable compliant access. Data owners must be able to control the use of their data for analytics purposes through an opt-in consent mechanism, with consent stored in a database.

Funded by the European Union

## iii.Establish confidence in data management and processing

To foster the confidence in data usage, tools and mechanisms are needed to define and enforce data sharing policies. This ensures that data is accessed and shared in a controlled manner and supports maintaining compliance with regulatory frameworks. Clear and comprehensive policies should be established. These policies must define the permissions for copying and storing data. The architecture should be configured to ensure data delivery aligning with agreed-upon terms. When dealing with federated data systems that incorporate components hosted on public infrastructure, specific policy considerations must be addressed. Access rights, including read, use, edit, and forward permissions, must be defined for specific data sets, Key Performance Indicators (KPIs), and reports. Data movement policies should be tailored to individual use cases, defining the type of analysis, the data involved, and the anticipated results. The system must implement data lifecycle management, encompassing data retention, overall management, and data sharing policies. The necessary technical infrastructure is  required to enforce these policies and ensure secure data handling. Data processing and storage must occur within designated, controlled infrastructure. Different computational methods should be made available. Technical components facilitating privacy-compliant data access, such as tools for anonymization, anonymity verification, and consent validation, are needed. Data sanitization methods, such as feedback mechanisms for data owners, should be provided. Data sanitization should be considered, where implementation details are tailored to each pilot. Finally, the security of data access during data exchange must be ensured.

Funded by the European Union

# 2. Trustworthy Architecture View

## Trustworthy Architecture

In the previous deliverable 5.2 [2], we described how different attestation mechanisms enable stakeholders to gain trust in TEADAL and the data-sharing process. In this iteration, we focus on component-specific descriptions and interactions that make up the trustworthy architecture of TEADAL.

TEADAL leverages established data mesh architectures but augments them with the notion of a federated data product which can be stretched across data lakes, thus ensuring seamless integration of tools and data products across heterogeneous environments in the computing continuum. However, in practice federated data products alone are not enough to effectively enable distributed, decentralized data exchange: the parties involved in the exchange process need to trust the process is carried out according to stipulated terms.

In the age of big data, safeguarding the integrity, confidentiality, and privacy of information is critically important. Traditional data pipelines, which handle the collection, processing, and analysis of vast amounts of data, often struggle with security and privacy issues. These challenges are particularly pronounced when it comes to distributing or scheduling tasks across various computational environments.

With the integration of advanced technologies, Privacy-Preserving Data Pipelines can maintain the highest standards of data security and privacy, even in complex and distributed computational environments. The following sections describe the first investigative efforts of integrating such PETs into pipeline workflow engines, both from an architectural and trust-modelling perspective and from a prototyping stance. The maturity of the work developed here is mainly dependent on the advancements of the baseline technologies and practical frameworks deployed.

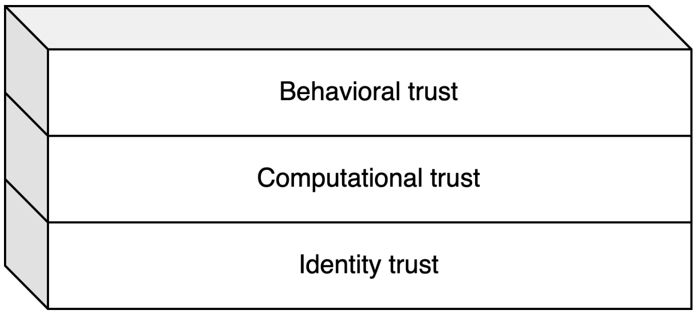


*Figure 1: TRUSTWORTHINESS LAYERS.*

As described in Figure 1, we characterize the foundational three layers that underpin trust in TEADAL: identity trust, which confirms the authenticity of entities; computational trust, which guarantees the integrity and confidentiality of computations; and behavioral trust, which assesses trustworthiness based on past actions and observed behavior.

Funded by the European Union

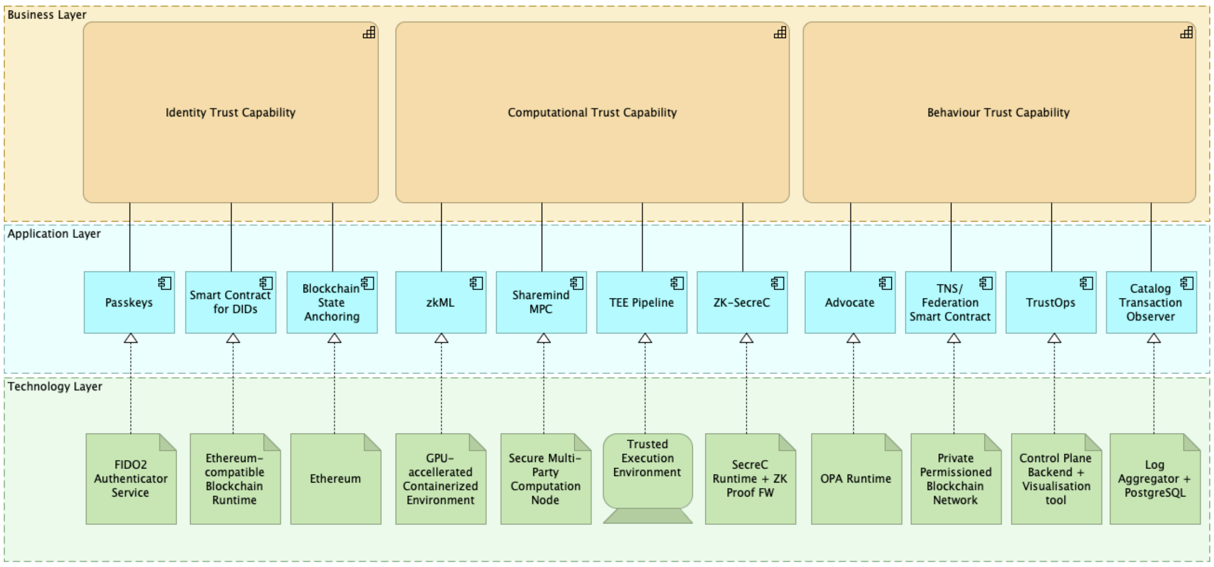

Business Layer
Identity Trust Capability
Computational Trust Capability
Behaviour Trust Capability
Application Layer
Passkeys
Smart Contract for DIDs
Blockchain State Anchoring
zkML
Sharemind MPC
TEE Pipeline
ZK-SecreC
Advocate
TNS/ Federation Smart Contract
TrustOps
Catalog Transaction Observer
Technology Layer
FIDO2 Authenticator Service
Ethereum-compatible Blockchain Runtime
Ethereum
GPU-accellerated Containerized Environment
Secure Multi-Party Computation Node
Trusted Execution Environment
SecreC Runtime + ZK Proof FW
OPA Runtime
Private Permissioned Blockchain Network
Control Plane Backend + Visualisation tool
Log Aggregator + PostgreSQL

*FIGURE 2: TEADAL TRUST PLANE OVERVIEW.*

Figure 2 shows how the different tools and components of the TEADAL Trust Plane establish trust across different layers of the stack. The Business Layer defines the three trust capabilities of identity trust, computational trust and behavior trust. The Application Layer houses specific tools and components of the Trust Plane, whereas the Technology Layer describes the enabling technologies for these tools and components: For Identity trust, it includes Passkeys, Smart Contracts for DIDs and Blockchain State Anchoring. For computational trust, it refers to zkML, Sharemind MPC, TEE Pipeline, and ZK-SecreC. Behavior trust comprises Advocate, TNS/Federation Smart Contract, TrustOps, and the Catalog Transaction Observer.

### i.Identity trust

Identity trust refers to the assurance that an entity is who they claim to be. It indicates that actions can be clearly attributed to the responsible parties. Identities should be established in an immutable and verifiable manner. An identity can be associated with the TEADAL data lake roles, such as the Data Lake Operator (DLO), FDP Designer, FDP Developer, and the FDP Consumer, but also the FDP itself, or any piece of software.

In TEADAL, we consider multiple ways to represent an identity: Either by using Decentralized Identifiers (DIDs), shared and federated namespaces, or through a variety of methods to automatically and indiscriminately collect evidence using Advocate and the Trust Plane Catalogue component integration. DIDs provide cryptographically verifiable identities without relying on a central authority. Shared and federated namespaces enable interoperability and consistent identity management.

Confirming or denying that a claimed identity is correct is achieved in multiple ways. The DLO, FDP Designer, FDP Developer, and the FDP Consumer, but also the FDP can be represented by a DID. Their identities are verifiable by resolving the associated DID document, which contains a description of the verification method. Typically, this provides means to verify ownership of a given keypair. For software, identity trust can be established by leveraging reproducible and verifiable infrastructure.

### ii.Computational trust

Funded by the European Union

In addition to identity trust, we must be able to guarantee trust in computation. This refers to the mechanisms that ensure the integrity and confidentiality of data and computations. The core objective is to establish verifiable security guarantees that prevent unauthorized access, tampering, or data leakage, thereby fostering confidence in the data-sharing processes in TEADAL.

To achieve this, different Privacy-Enhancing Technologies (PETs) can be employed. Trusted Execution Environments (TEEs) create a secure enclave within a processor. This ensures that sensitive data is processed in isolation from the rest of the system, protecting it from unauthorized access, even if the operating system or other applications are compromised. Secure Multi-Party Computation (MPC) enables multiple parties to collaboratively compute functions over their respective inputs while keeping those inputs private. This allows for confidential data processing without exposing sensitive information to any of the involved parties. Zero-Knowledge Proofs (ZKP) provide a way for one party to prove the truth of a statement to another without revealing any additional information beyond the statement’s validity, enhancing verification without compromising privacy. Additionally, we integrate Synthetic Data Generation methods to produce anonymized datasets that retain key statistical properties, enabling secure testing and analysis without exposing sensitive details.

#### iii.Behavioral trust

Behavioral trust is trust based on an entity's observed actions and history. It relies on the analysis of past behavior to predict future trustworthiness. Behavioral trust mechanisms evaluate a variety of factors, including policy compliance, consistency of actions, or historical interactions. These assessments help determine whether an entity can be trusted to operate within a given system.

Behavioral trust goes beyond a static evaluation and takes into account data and interactions in real time: An entity that demonstrates a stable pattern of trustworthy behavior builds trust within the system. If an entity shows unexpected behavior, its trustworthiness is adjusted accordingly. In TEADAL, behavioral trust is typically related to reputation-based mechanisms that aggregate the historical behavior of entities and make this information available to others. This reputation is determined by the long-term track record of an entity's actions within the system.

### B. Components Description and Functions

This section provides an overview of the components and functionalities of the TEADAL Trust Plane by focusing in particular on the three trust layers of identity, computational and behavioral trust. Therefore we explore blockchain state anchoring, Decentralized Identifiers and Passkeys, and privacy-preserving technologies like Zero-Knowledge Machine Learning, Multi-Party Computation, and Trusted Execution Environments. Furthermore, we examine the role of Advocate, the Teadal Name Service, and Federation Smart Contracts in ensuring verifiable operations and fostering behavioral trust.

#### i.Blockchain State Anchoring

To improve private blockchain integrity and immutability we have implemented a protocol that periodically anchors the private blockchain state to public blockchains, taking advantage of the security features of larger, decentralized networks.
The proposed protocol involves periodically creating a snapshot of the private blockchain's state, represented by a cryptographic hash, at specified intervals. This snapshot is then recorded onto a public blockchain, ensuring a tamper-resistant record of the private

blockchain's state at various points in time. In case of suspected tampering, participants can verify the private blockchain's integrity by comparing its current state with the previously anchored snapshots on the public blockchain.

Any TEADAL component that use a blockchain, could use or deploy smart contracts on a private permissioned blockchain, increasing:

- Security: only approved actors can write on the permissioned blockchain
- Privacy: the information stored on the blockchain are not exposed publicly
- Throughput: usually, a private permissioned blockchain can handle more transaction per seconds than a public ones

Also, the component doesn't have to handle the cryptocurrency costs, as the private blockchain can handle gas free transactions or fund with private token the component. Private blockchain could:

- Do not expose endpoints directly and/or gain write rights only to selected actors/components and use a gas free configuration, meaning that only privileged actors/components that own a privileged account can participate in the network without worrying about wallet liquidity.
- Use a private token, funding only certains actors/components and asking for a fee for each transaction, meaning that only funded accounts can participate in the network.

Private blockchain lacks of security, as the number of the validators are not high as public networks, but having a periodic snapshot anchored to one or multiple public networks, increase the security, as the snapshot is a cryptographic hash that any participant (usually validators) can verify, and if one validator is malicious (E.G. it tries to anchor a different state and diverge the private blockchain history) it can be slashed or even excluded as a validator.

## ii.Passkeys

Passkeys are cryptographic credentials based on a public-private key pair. The private key is securely stored by the user, while the public key is used for verifying digital signatures. Thanks to their cryptographic nature, passkeys eliminate risks associated with weak or compromised passwords.

Passkeys can be implemented using various signature curves, depending on security needs and the technical requirements of the environment.

In EVM-compatible environments, the secp256K1 elliptic curve is commonly used for passkeys. This curve is widely adopted in the blockchain ecosystem.

Passkey registration involves generating a pair of cryptographic keys: a private key (securely stored on the device) and a public key (sent to the server). The server sends a unique challenge, which the user signs with their private key to verify their identity. The public key, along with metadata, is stored on the server for future authentications.

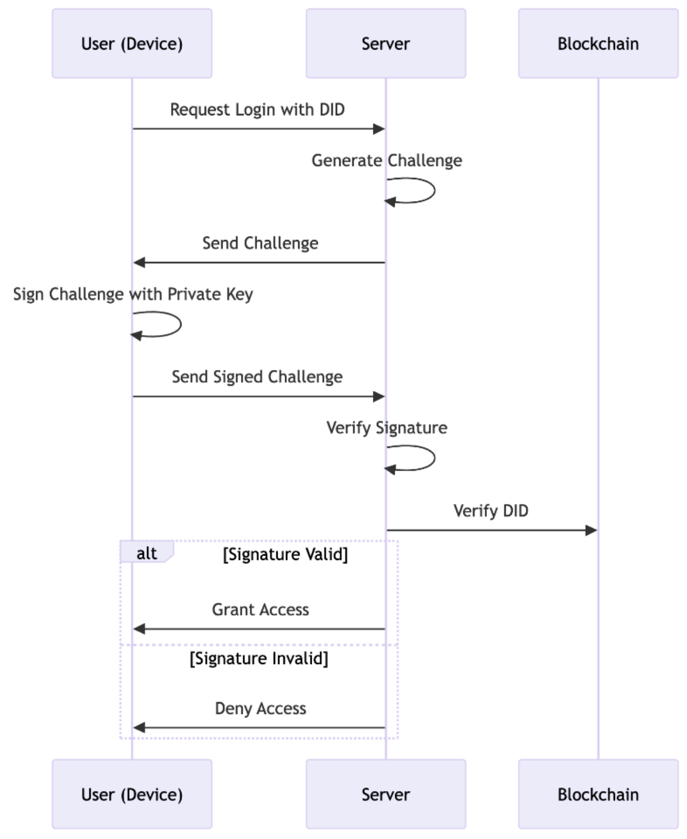
**Benefits of Public Key and DID Association:**

- Interoperability: DIDs can be used across multiple platforms, enabling portable and universal identities. This simplifies and scales identity management.
- DID Updates: In case of private key compromise, the DID can be updated with a new public key, ensuring continuity of identity.

Funded by the European Union

- Advanced Features: DIDs support the integration of verifiable credentials, allowing the addition of metadata or attestations linked to the user's identity without compromising privacy

The verification workflow leverages asymmetric cryptography and blockchain to ensure secure, decentralized, and passwordless authentication. In this system, the user owns a private key associated with a unique DID that is registered on the blockchain. The login process is based on a challenge signed with the user's private key, verified through the corresponding public key. The server can then verify on the blockchain to confirm the validity of the associated DID.
If the signature is valid and the DID matches, access is granted. Otherwise, access is denied, ensuring a secure, decentralized, and passwordless process.



## iii. Smart Contract for DIDs

Decentralized Identifiers (DIDs) represent an emerging paradigm in the field of digital identity. Based on decentralized technologies such as blockchain, DIDs ensure security, integrity, and interoperability across a wide range of applications.

Their functionality at the Smart Contract level enables autonomous and verifiable management of digital identities, eliminating reliance on centralized authorities. Thanks to advanced cryptographic mechanisms, DIDs offer a scalable and censorship-resistant model, fostering a more secure and reliable ecosystem.

The use of DIDs allows users to maintain full control over their credentials, reducing the risk of identity theft and third-party abuses. Furthermore, their interoperability facilitates integration with various platforms and systems, promoting a global and decentralized digital identity.

Funded by the European Union

A Decentralized Identifier (DID) system is designed to ensure security, autonomy, and verifiability in digital identity management. Its architecture leverages blockchain technology to provide a trustless, tamper-resistant framework where users maintain control over their identities. The following sections explore the key aspects that make this system robust and reliable.

The DID Registry is a Smart Contract that acts as a decentralized registry to manage operations on DIDs. Thanks to the blockchain, the registry ensures:

- Immutability: Recorded information cannot be retroactively altered.
- Transparency: Every operation is visible on the blockchain.

The lifecycle of a DID is divided into four main stages:

- Creation: Generation of a public/private key pair and registration of the DID on the blockchain. This operation is performed by the DID Issuer, who associates a public key with the registered DID.
- Resolution: Retrieval of the DID Document through a resolver. A resolver interprets the DID and returns the associated document.
- Update: Modification of the information. This operation allows, for example, the updating of public keys or changing the associated services.
- Revocation: Disabling the DID or removing the information from the registry, ensuring it can no longer be used.

The provided Smart Contract implements all the key functionalities required to manage DIDs. Below, the main sections of the code and their respective concepts are analyzed. The constructor of the Smart Contract is used to initialize the decentralized identity registry. Example:

```
constructor(string memory _identifier) {
     _scDidPrefix = string(
     abi.encodePacked(
               DidUtils.DID_PREFIX,
               Strings.addressToString(address(this)),
               ':'
     )
     );

     string memory selector = Strings.bytesToHexWithoutPrefix(
     abi.encodeWithSelector(
               DecentralizedID.validateDecentralizedId.selector
     )
     );

     _verificationMethodDid = string(
     abi.encodePacked(_scDidPrefix, selector)
     );

     _providerId = _generateId(_identifier);
     string memory did = string(
     abi.encodePacked(_scDidPrefix, _providerId)
     );

     _provider = Identity({
     owner: msg.sender,
     did: did,
     identifier: _identifier
     });
}
```

This constructor:

Funded by the European Union

1. Sets the DID prefix based on the Smart Contract address
2. Generates a DID verification method
3. Creates an initial identity for the provider (the contract deployer)

The *addIdentity* method allows adding a new decentralized identity. This is the starting point for issuing a DID:

```
function addIdentity(string memory _identifier, address _account) external onlyOwner {
    require(_account != address(0x0), DidUtils.INVALID_ADDRESS);

    bytes memory checkExist = bytes(_owners[_account]);
    require(checkExist.length == 0, DidUtils.DID_ALREADY_EXISTS);

    string memory id = _generateId(_identifier);
    string memory did = string(abi.encodePacked(_scDidPrefix, id));
    bytes32 bytesId = keccak256(abi.encodePacked(id));

    _entities[bytesId] = Identity({
    owner: _account,
    did: did,
    identifier: _identifier
    });

    _owners[_account] = id;
    emit NewIDentity(did, _identifier, block.timestamp);
}
```

This method performs the following operations:

3. Verifies that the account does not already have an associated DID.
4. Generates a new unique identifier using *_generateId*.
5. Registers the identity in the *_entities* mapping and associates the DID with the provided account.

It emits a *NewIDentity* event to notify the creation of the new DID.

The Smart Contract includes functionality to track the ownership history of decentralized identities. This feature is implemented through the *_ownershipHistory* mapping.

The *_ownershipHistory* mapping uses a two-level structure:

```
mapping(bytes32 => mapping(address => uint256)) private _ownershipHistory;
```

- The main key is a hash of the DID.
- The secondary key is the owner's address.
- The stored value is a timestamp indicating when the owner transferred or lost the identity.

When an identity is transferred, the contract records the change in the history:

```
  function updateOwnerIdentity(
    string calldata _did,
    address _new,
    address _current
  ) external override onlyOwner {
    string memory did = getDecentralizedId(_current);
    require(_new != address(0x0), DidUtils.INVALID_ADDRESS);
    require(Strings.equal(did, _did), DidUtils.INVALID_DID);

    (string memory scDidPrefix, string memory id) = DidUtils.splitDid(
    _did
    );
    require(
    Strings.equal(_scDidPrefix, scDidPrefix),
    DidUtils.INVALID_DID
    );
```

Funded by the European Union

```
    bytes memory checkExistOld = bytes(_owners[_current]);
    require(checkExistOld.length != 0, DidUtils.ADDRESS_NOT_FOUND);

    bytes memory checkExistNew = bytes(_owners[_new]);
    require(checkExistNew.length == 0, DidUtils.DID_ALREADY_EXISTS);

    bytes32 bytesId = keccak256(abi.encodePacked(id));
    require(
    _entities[bytesId].owner != address(0x0),
    DidUtils.DID_NOT_FOUND
    );

    delete _owners[_current];

    _entities[bytesId].owner = _new;
    _owners[_new] = id;

    _ownershipHistory[bytesId][_current] = block.timestamp;
  }
```

It is possible to verify if an account was a past owner of a DID by comparing the *_ownershipHistory* mapping with the required address and timestamp.

The ownership history enhances the transparency, security, and reliability of property and credential management systems.

- Audit and Verification: Enables tracking of all property changes, which is useful for corporate or security audits
- Dispute Resilience: Allows the legitimacy of past property ownership to be demonstrated
- Credential Persistence: Verifiable Credentials remain valid even if public keys are updated. The signature associated with the old public key ensures that no previously issued credential becomes invalid, maintaining the system's reliability and consistency.

To ensure that the issuance, update, or revocation of identities is approved by multiple entities, a multisig (multi-signature) wallet can be integrated. A multisig wallet is a type of wallet that requires the consent of multiple private key holders to authorize a transaction, thereby distributing control and reducing the risk of unilateral actions. This can be implemented by adding checks requiring the approval of a quorum of administrators.

Example function for multisig approvals:

```
address[] public owners;
mapping(address => bool) public isOwner;
uint public numConfirmationsRequired;
```

```
struct Transaction {
    address to;
    uint value;
    bytes data;
    bool executed;
    uint numConfirmations;
}
modifier onlyOwner() {
    require(isOwner[msg.sender], "not owner");
    _;
```

Funded by the European Union

```
}
function submitTransaction(address _to, uint _value, bytes memory _data) public onlyOwner {
    uint txIndex = transactions.length;

    transactions.push(Transaction({
    to: _to,
    value: _value,
    data: _data,
    executed: false,
    numConfirmations: 0
    }));

    emit SubmitTransaction(msg.sender, txIndex, _to, _value, _data);
}

function    confirmTransaction(uint    _txIndex)    public    onlyOwner    txExists(_txIndex)
notExecuted(_txIndex) notConfirmed(_txIndex) {
    Transaction storage transaction = transactions[_txIndex];
    transaction.numConfirmations += 1;
    isConfirmed[_txIndex][msg.sender] = true;

    emit ConfirmTransaction(msg.sender, _txIndex);
}
function    executeTransaction(uint    _txIndex)    public    onlyOwner    txExists(_txIndex)
notExecuted(_txIndex) {
    Transaction storage transaction = transactions[_txIndex];

    require(transaction.numConfirmations >= numConfirmationsRequired, "cannot execute tx");

    transaction.executed = true;

    (bool success, ) = transaction.to.call{value: transaction.value}(transaction.data);
    require(success, "tx failed");

    emit ExecuteTransaction(msg.sender, _txIndex);

}
```

With this structure:

- Administrators can approve actions such as DID creation or revocation.
- An action is executed only when the number of approvals exceeds the defined quorum.

### i.Zero-Knowledge Machine Learning

Zero-Knowledge Machine Learning (ZKML) is a cryptographic approach that enables the verification of machine learning model inferences on blockchain networks without exposing the underlying data or computations. This innovative technology facilitates secure, privacy-preserving, and transparent use of AI models in decentralized applications, ensuring the integrity and trustworthiness of the results.

Funded by the European Union

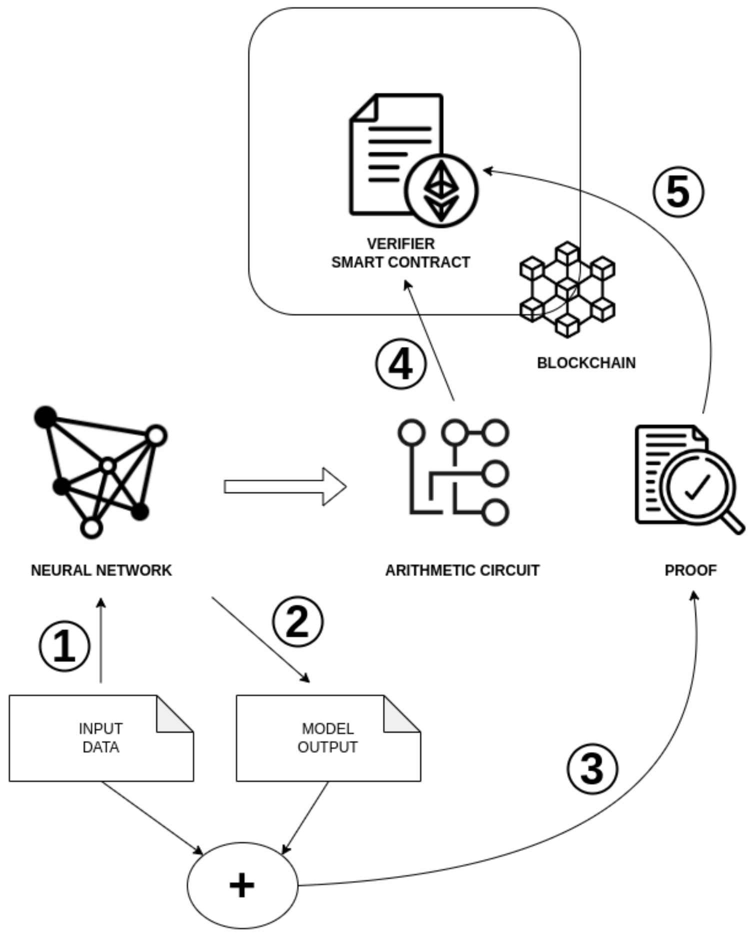


FIGURE 4: ZERO-KNOWLEDGE MACHINE LEARNING (ZKML) OVERVIEW.

Easy Zero-Knowledge Inference (EZKL) is a powerful library designed to simplify inference verification using Zero-Knowledge Proofs (ZKPs) and blockchain technology. It enables developers to integrate AI model verifications into blockchain ecosystems with minimal complexity.

The EZKL library combine the following elements:

- An AI Machine Learning Model: the trained model to be used for inference.
- A Circuit Compatible with zk-SNARKs: the arithmetic circuit that represents the model's computation steps in a zero-knowledge format.
- An EVM-Compatible Blockchain: the decentralized network where the verifier can trustlessly validate the inference proof.

ZKML could provide significant advantages. Integrating it with a TEADAL AI model we can achieve:

- Enhance data security: input data does not need to be shared to verify the inference process
- Introduce trustlessness: decentralized inference verification through smart contract can be validated by any third party without relying on a central authority

The steps to generate a verifiable zk-proof for each inference process can be automated, after having identified the target model, the steps are:

Funded by the European Union

1. Settings generation:
    a. The EZKL library generates a settings file containing information about the model and the desired visibility of input/output parameters.
    b. Parameters can be set as public, hashed, or private. For privacy-preserving purposes, input parameters are typically hashed or private.
2. Circuit Compilation:
    a. The neural network is compiled into an arithmetic circuit compatible with zk-SNARK proof systems.
    b. Each computational step of the network is translated into polynomial equations that describe the relationships between inputs, intermediate computations, and outputs.
3. Environment setup:
    a. Since the EZKL library employs ZK-SNARK proof construction techniques, the proof environment requires a setup ceremony. This can be securely accomplished using the SRS (Structured Reference String) from the Perpetual Power of Tau [6], a trusted setup protocol.
4. Setup and keys generation:
    a. EZKL executes the necessary trusted setup and generates the Prover key, used to create zk-proofs for inference processes, and the Verifier key, used to validate these proofs.

Then, with EZKL, we will be able to generate the Verifier Smart Contract, where the Verifier key is embedded in the contract itself.

This smart contract could be deployed on a TEADAL permissioned blockchain or any EVM compatible network.

Once deployed, anytime an inference process is done using the same AI model, a zk-proof of this process can be generated by EZKL and this proof can be verified by anyone using the already deployed smart contract.

### ii.Advocate

Advocate collects, combines, and stores evidence from multiple sources in the TEADAL ecosystem. By ensuring that all claims are substantiated with telemetry data, Advocate provides an overview of the data lake's operations. This holistic approach allows for more effective monitoring, auditing, and compliance, helping organizations meet regulatory requirements and maintain trust with consumers of FDPs. These claims range from those representing service deployments within a cluster, e.g., FDP creation. In Catalogue-level, claims are a report about consumer interactions, e.g., when creating a contract for consuming an FDP. These claims are stored along with tracing data from observation tools that substantiate them. Observation tools provide tracing logs and metrics, generating Platform-level evidence that Advocate collects, combines, and immutably stores to justify claims. Tracing logs are also used to provide evidence about data exchanges between services. Moreover, we aim to create reliable and trustworthy monitoring sources by enabling mTLS and signed deployment images for all critical infrastructure services.

The architecture of Advocate is designed to handle a diverse range of evidence sources and types, e.g., Jaeger and Kubernetes. By supporting various evidence sources from different providers and integrating seamlessly with existing infrastructure of data lakes, Advocate ensures that the integrity of evidence is maintained across the TEADAL environment and across all data lakes. The use of private keys for signing evidence adds an additional layer of authenticity, making it difficult for unauthorised parties to alter or forge claims alongside using a blockchain for tamper proof record keeping.

Funded by the European Union

In a federated environment like TEADAL, where every data lake has its own instance of Advocate running, we need to aggregate all the evidence and verifications from different instances across the data lakes. To achieve this, each Advocate instance in each data lake acts as a gateway and provides evidence to other data lakes. This way other Advocate instances can discover, verify and enrich observations and combine evidence to build up a comprehensive picture of FDP, sFDP data exchanges. Moreover, since Advocate uses the InterPlanetary File System (IPFS) as shared storage, and a public (or accessible) blockchain with smart contracts, other third-party implementations can interact and use the evidence created by Advocate across the TEADAL data lakes if needed.

As part of the trustworthy architecture of TEADAL, Advocate plays a crucial role in ensuring identity trust, computational trust, and behavioral trust. To uphold identity trust, Advocate assigns and manages public/private keys to establish identities of TEADAL components, such as FDPs. In fulfilling computational trust, Advocate guarantees that all interactions within TEADAL remain transparent and tamper-proof. By maintaining immutable storage for logs and traces, Advocate ensures that every action can be traced back to its origin, securing data integrity and preventing malicious alterations. For behavioral trust, Advocate systematically collects and records claims at both the application and platform levels. By storing signed claims and validating them against predefined policies and regulations, Advocate ensures that actions within TEADAL comply with established rules. Furthermore, Advocate facilitates attestation by providing a transparent infrastructure where FDP policies and supporting evidence are stored, allowing data consumers and providers to verify compliance. The ability to trace DIDs through recorded evidence further strengthens confidence in the federation's trust model.

Figure 5 illustrates the claim verification process between two Advocate instances, Advocate A and Advocate B, which can be divided into two distinct phases: Key Generation and Claim Verification:

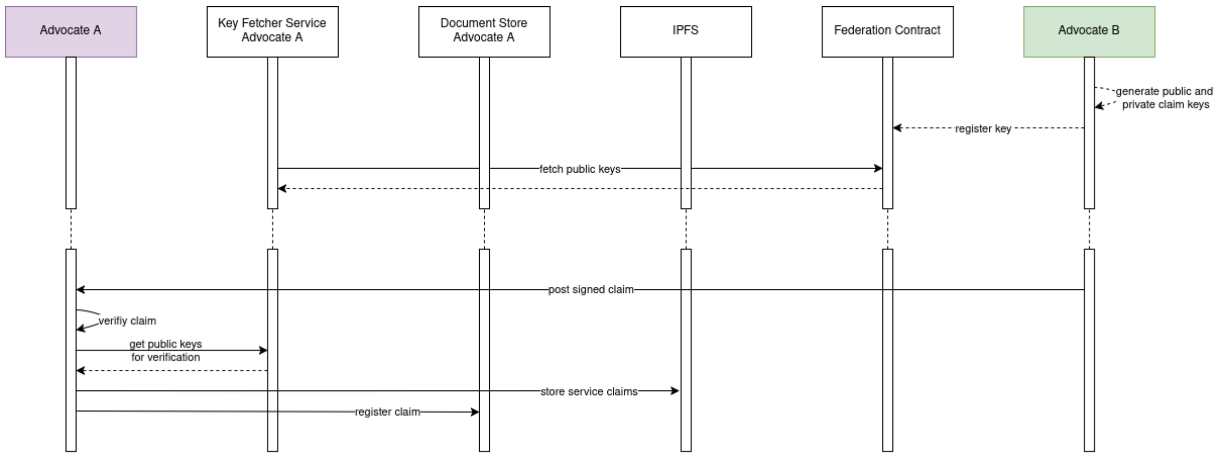


*FIGURE 5: CLAIM VERIFICATION BETWEEN TWO ADVOCATE INSTANCES.*

The key generation phase consists of creating and registering the necessary cryptographic material for later claim verification. In this phase, Advocate B first generates a public/private key pair and registers the public key in the Federation Contract. Following this registration, Advocate A's Key Fetcher Service retrieves this public key from the Federation Contract and stores it for later use. Based on that, in the Claim Verification phase, a signed claim is exchanged and verified. It is assumed that Advocate B has signed a claim using its previously registered key pair. This signed claim is then sent to Advocate A. Upon receipt, Advocate A verifies the claim by retrieving the corresponding public key using its Key Fetcher service. If the cryptographic verification is successful, the service claims associated with the

Funded by the European Union

verified claim are stored in the IPFS, and a record of the claim is registered in Advocate A's Document Store smart contract.

### iii. TNS and Federation Smart Contract

The Teadal Name Service (TNS) and Federation Smart Contracts represent two technologies to build trust in identities within decentralized systems like TEADAL. They utilize smart contracts running on a distributed ledger technology (DLT) to provide enhancements for evidence generated by Advocate.

The TNS serves as a decentralized naming system utilizing core functionalities of the Ethereum Name Service (ENS) smart contracts. Analogously to a classical Domain Name Service, the TNS records federation members by providing an abstraction layer to associate human-readable names with blockchain accounts or resources. Moreover, TNS creates resolvable names for both Federated Data Products (FDPs) and Shared Federated Data Products (SFDPs), streamlining resource discovery and access within the ecosystem. This enhances the usability and accessibility of the TEADAL Federated Data Lakes for all federation members.

The Federation Smart Contracts manage governance and operations of the federation by defining an initial set of members and establishing DLT-backed protocols for registering new members in a collaborative manner. They provide mechanisms for federation members to register domains and services in the TNS and enable them to register public keys, facilitating trustworthy and secure verification among participants.  Any modifications to the federation are logged in an immutable manner, by emitting events in the smart contract. The events are related to membership management. They encompass events such as MemberAdded, MemberApproved, and NewApprovalCount. This provides a transparent and verifiable history of federation governance.

The TNS and Federation Smart Contracts extend Decentralized Identifiers (DIDs) to establish DLT-backed and self-sovereign digital identities, building a strong foundation for trustworthy digital identities in TEADAL. This framework is important for generating and verifying evidence within decentralized systems, supporting both day-to-day operations and building a foundation of trust and accountability.

The ENS/Federation Smart Contracts provide decentralized mechanisms for facilitating interactions between federations. The TEADAL Name Service (TNS) serves as a decentralized naming system within the distributed ledger technology (DLT) where TEADAL operates, with more functionalities analogously to classical Domain Name Service to do records management at a federation members level. In conjunction with TNS, Federation Smart Contracts play a role in managing the governance and operations of the federation. Building on the previous iterations of TEADAL and the TEADAL Trust Plane, the ENS/federation smart contracts now enable interaction with multiple Advocate instances across federations.

Figure 6 shows the interaction between two Advocate instances for validating changes to an sFDP using the Document Store Contract. The Document Store Contract serves as the single source of truth for registered claims:

Funded by the European Union

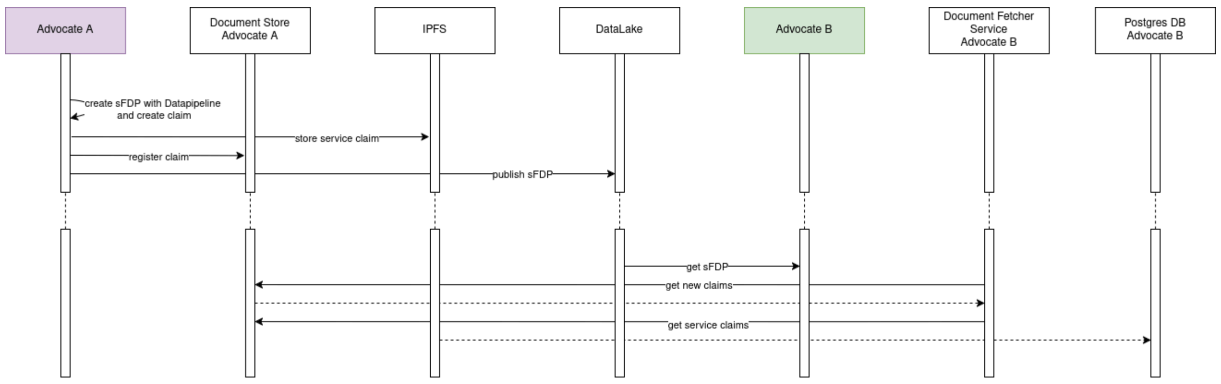


*FIGURE 6: SFDP UPDATE BETWEEN TWO ADVOCATE INSTANCES.*

Each step of the data pipeline is documented by Advocate by generating signed claims. Therefore, Advocate A first creates a new claim that includes data from an sFDP. This claim is stored in the IPFS, and a corresponding registration of the claim is recorded in Advocate A's Document Store smart contract. Upon successful registration, the sFDP is published to the DataLake. Next, Advocate B uses the sFDP and retrieves the associated claim using the Document Fetcher Service. It retrieves the previously recorded registration data from Advocate A's Document Store Smart Contract and initiates storage of the claim content retrieved from IPFS in a local database of Advocate B.

To strengthen behavioral trust, the Federation Smart Contract incorporates a reputation and dispute resolution system. If conflicts arise, the smart contract logs challenges and enforces resolution mechanisms, ensuring fair and transparent outcomes. Reputations are recorded on-chain. Currently, they are defined by tracking disputes relating to a dataset. If an identity has multiple disputes assigned to it, its reputation is affected.

### iv.Catalog transaction Observer

The interactions between users and the Catalogue fall into two categories: browsing for assets (datasets, FDPs, SFDPs) and contributing to the catalogue itself. This last category of interactions is the target of the Transactions Observer implementation. The Catalogue can be configured to execute BPMN processes as a response to two main interactions:

- lifecycle events (publishing, editing, deleting) related to assets descriptions
- requests that a user can trigger related to a specific asset, like requesting access to an FDP or requesting to advertise an asset through a dataspace connector

Inserting BPMN activities in the right parts of such processes allow immutability logging events to Advocate.

In case of a lifecycle event, the following process represents an example of where to add such an activity

Funded by the European Union

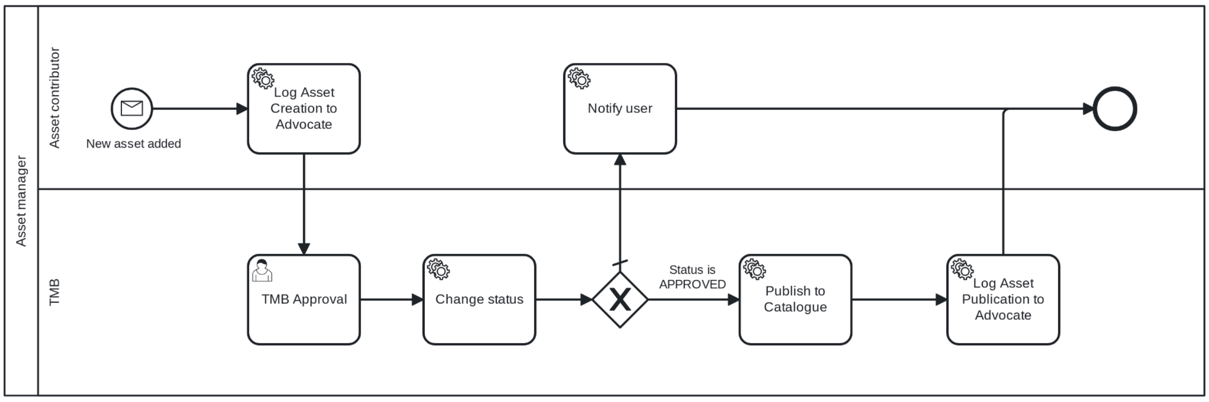
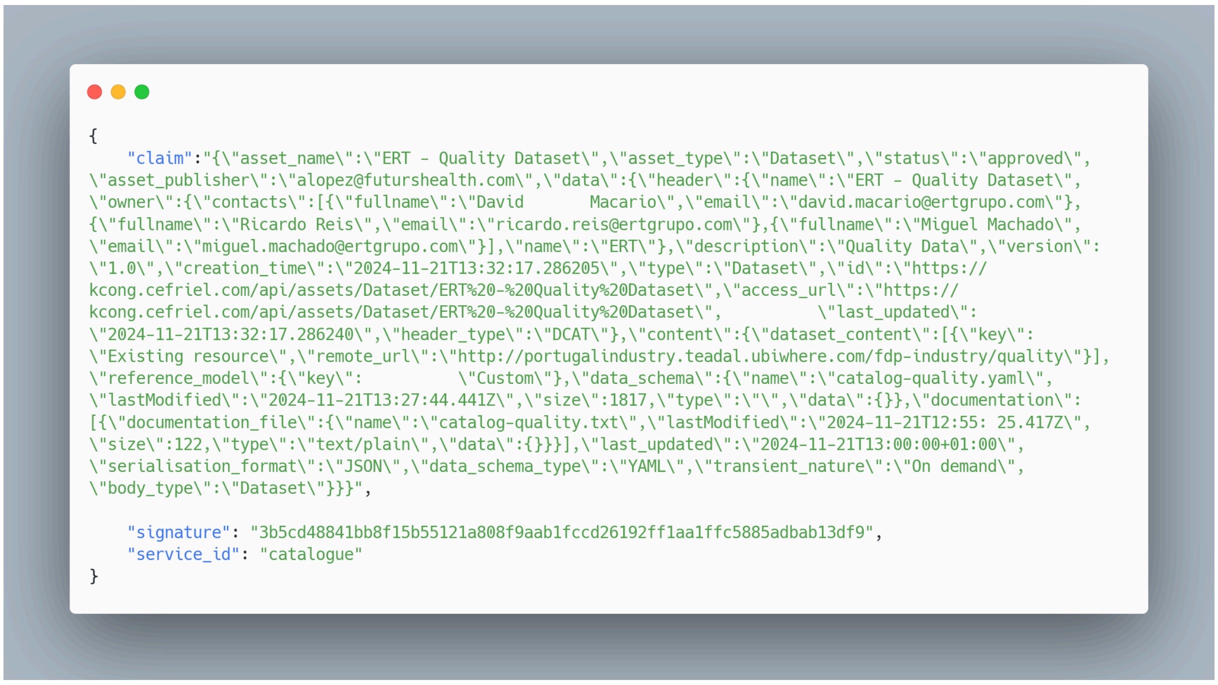


FIGURE 7: ADVOCATE LOGGING THROUGH BPMN SERVICE TASKS.

In details, the activity is a Service Task that performs an API call directed to Advocate. An example of a lifecycle event sent to Advocate is listed below:



```
{
    "claim":"{\"asset_name\":\"ERT - Quality Dataset\",\"asset_type\":\"Dataset\",\"status\":\"approved\",
\"asset_publisher\":\"alopez@futurshealth.com\",\"data\":{\"header\":{\"name\":\"ERT - Quality Dataset\",
\"owner\":{\"contacts\":[{\"fullname\":\"David       Macario\",\"email\":\"david.macario@ertgrupo.com\"},
{\"fullname\":\"Ricardo Reis\",\"email\":\"ricardo.reis@ertgrupo.com\"},{\"fullname\":\"Miguel Machado\",
\"email\":\"miguel.machado@ertgrupo.com\"}],\"name\":\"ERT\"},\"description\":\"Quality Data\",\"version\":
\"1.0\",\"creation_time\":\"2024-11-21T13:32:17.286205\",\"type\":\"Dataset\",\"id\":\"https://
kcong.cefriel.com/api/assets/Dataset/ERT%20-%20Quality%20Dataset\",\"access_url\":\"https://
kcong.cefriel.com/api/assets/Dataset/ERT%20-%20Quality%20Dataset\",          \"last_updated\":
\"2024-11-21T13:32:17.286240\",\"header_type\":\"DCAT\"},\"content\":{\"dataset_content\":[{\"key\":
\"Existing resource\",\"remote_url\":\"http://portugalindustry.teadal.ubiwhere.com/fdp-industry/quality\"}],
\"reference_model\":{\"key\":          \"Custom\"},\"data_schema\":{\"name\":\"catalog-quality.yaml\",
\"lastModified\":\"2024-11-21T13:27:44.441Z\",\"size\":1817,\"type\":\"\",\"data\":{}},\"documentation\":
[{\"documentation_file\":{\"name\":\"catalog-quality.txt\",\"lastModified\":\"2024-11-21T12:55: 25.417Z\",
\"size\":122,\"type\":\"text/plain\",\"data\":{}}}],\"last_updated\":\"2024-11-21T13:00:00+01:00\",
\"serialisation_format\":\"JSON\",\"data_schema_type\":\"YAML\",\"transient_nature\":\"On demand\",
\"body_type\":\"Dataset\"}}}",

    "signature": "3b5cd48841bb8f15b55121a808f9aab1fccd26192ff1aa1ffc5885adbab13df9",
    "service_id": "catalogue"
}
```

FIGURE 8: PAYLOAD FOR LOGGING THE REGISTRATION OF A NEW DATASET.

User requests are data structures in the Catalogue allowing a ticket-like system. Once a User request is created, its status is tracked in the Catalogue and the corresponding BPMN process is started. The last activity of the BPMN process will be a call to the Catalogue API signalling that the User request can be “closed” since it was correctly handled. The implementation of the event logging to Advocate follows what was described before, as shown in the process below that corresponds to the User request for asking the permission to use an FDP:

Funded by the European Union

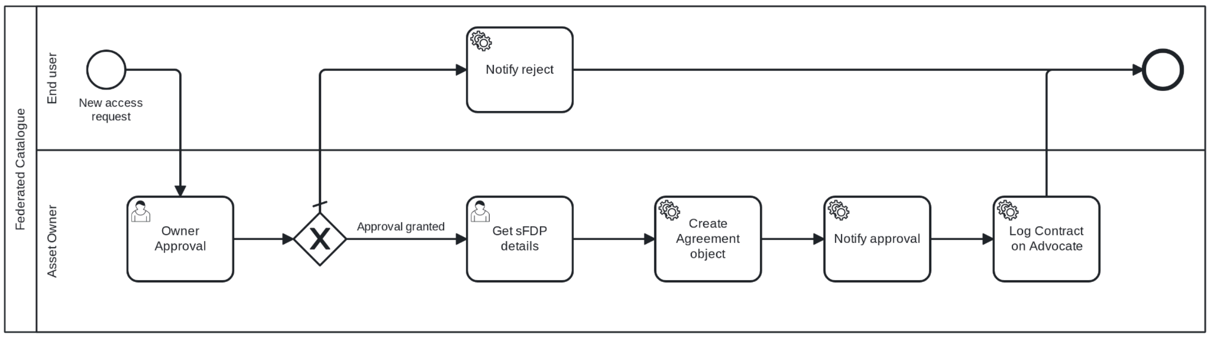
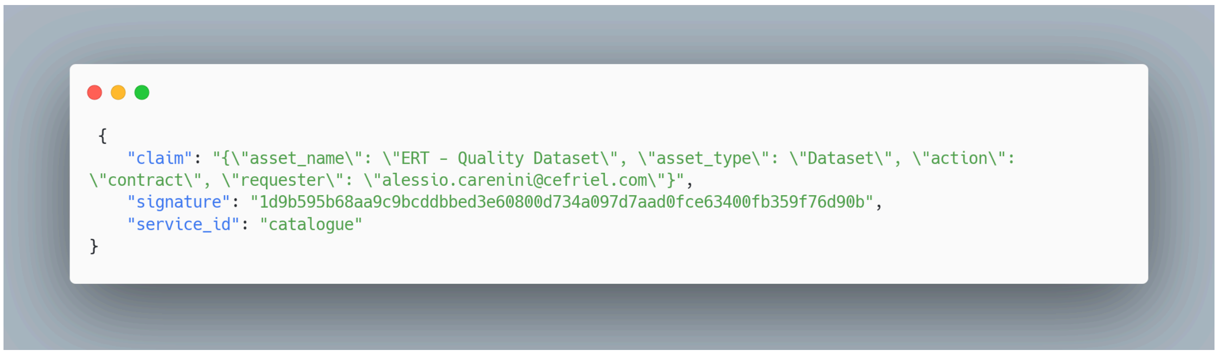


FIGURE 9: ADVOCATE LOGGING OF USER REQUESTS.

In this case, the body of the message sent to Advocate will be like the example below:



```
 {
    "claim": "{\"asset_name\": \"ERT - Quality Dataset\", \"asset_type\": \"Dataset\", \"action\":
\"contract\", \"requester\": \"alessio.carenini@cefriel.com\"}",
    "signature": "1d9b595b68aa9c9bcddbbed3e60800d734a097d7aad0fce63400fb359f76d90b",
    "service_id": "catalogue"
}
```

FIGURE 10: PAYLOAD FOR ADVOCATE LOGGING THROUGH BPMN SERVICE TASKS.

## v.Sharemind MPC

Sharemind MPC[1] is a secure multi-party computation (MPC) framework that enables multiple parties to jointly compute functions over their private data without exposing the underlying inputs. Using secret sharing techniques, sensitive data is divided into multiple shares and distributed among non-colluding nodes. In our implementation, we use the SecreC language[2], a high-level, domain-specific language for writing secure computation protocols, to define and execute these MPC protocols efficiently. These nodes then collaboratively perform computations on the shares and later reconstruct the final result, ensuring that no single party gains access to the complete dataset. This method provides robust privacy guarantees and complies with strict data protection standards. More details on MPC cryptographic mechanisms and Sharemind MPC tooling are in D5.2 [2].

Within TEADAL, Sharemind MPC is integrated into data pipelines to facilitate secure, collaborative data analysis and aggregation. The MPC runtime is deployed as an independent executable task, which can be orchestrated using Kubernetes to manage parallel execution across distributed nodes. Leveraging Kubernetes features such as scheduling and node affinity, MPC tasks are coordinated to perform distributed Privacy-Preserving Data Pipelines at scale. This integration not only enhances data confidentiality but also supports regulatory compliance and reinforces the trustworthiness of federated data processing operations.

[1] https://sharemind.cyber.ee/sharemind-mpc/
[2] https://docs.sharemind.cyber.ee/sharemind-mpc/2023.09/development/secrec-tutorial.html

Funded by the European Union

The Sharemind MPC runtime serves as a potential type of Privacy-preserving Data Pipeline task for distributed and collaborative computations. Leveraging secret sharing techniques, it allows multiple parties to jointly compute functions over private data without revealing the underlying inputs. Deployed as containerized tasks orchestrated by Kubernetes, the Sharemind MPC runtime ensures synchronized execution across non-colluding nodes, enabling secure data aggregation and analysis. Moreover, its integration with TEADAL's trust infrastructure—through evidence logging and auditing—enhances overall data privacy while supporting collaborative processing in federated environments.

### vi. TEE Pipeline

Trusted Execution Environments (TEEs) are hardware-based isolated areas within a processor that securely execute sensitive code and handle confidential data, even if the host operating system is compromised. By leveraging features such as memory encryption and secure boot, TEEs ensure data integrity and confidentiality during computation. As presented in D5.2 [2], TEEs like Intel SGX, Intel TDX, AMD SEV, and Arm TrustZone provide robust security guarantees that protect sensitive workloads from external tampering or unauthorized access. This technology enables applications to process private data securely by isolating critical operations from the rest of the system.

TEE-based tasks can be seamlessly integrated into data pipelines and orchestrated using container orchestration platforms like Kubernetes. For instance, lightweight virtual machines provided by technologies such as Kata Containers and Confidential Containers enable the deployment of TEEs in containerized environments, ensuring that sensitive computations run within secure enclaves. These solutions support essential features such as remote attestation, key management, and secure image decryption, which are critical for verifying the integrity of the TEE and the workload it hosts. By combining these capabilities with Kubernetes' scheduling, node affinity, and resource management, we can orchestrate energy-efficient, Privacy-Preserving Data Pipelines that meet both security and operational requirements.

Within TEADAL, Trusted Execution Environments (TEEs) provide an extra runtime mechanism for executing Privacy-preserving Data Pipeline tasks securely. By running sensitive computations inside hardware-isolated enclaves, TEEs ensure that data remains confidential even when operating on less trusted infrastructure. Integrated with Kubernetes through solutions like Kata Containers and Confidential Containers, TEE-based tasks can be dynamically scheduled based on resource availability and energy efficiency criteria. These secure runtime environments can also interact with TEADAL's identity management, observability, and evidence collection tools, reinforcing trust and ensuring that sensitive operations are verifiable and compliant with regulatory standards. A prior demonstration of TEE integration was detailed in D5.2 [2].

### vii. ZK-SecreC

ZK-SecreC[3] is a high-level, domain-specific language designed specifically for developing zero-knowledge proofs. Partially developed within the TEADAL project, it addresses the challenge of creating verifiable proofs while ensuring that sensitive data remains confidential. With a syntax reminiscent of modern languages like C++ or Rust, ZK-SecreC introduces an information flow type system that clearly delineates between public and private data. This design allows developers to write proof statements at a high level, while the language's compiler translates these into efficient low-level arithmetic circuits compatible with various ZK back-ends. Its rich library of reusable functions and gadgets further streamlines the development process, reducing complexity and enhancing reliability.

[3] https://github.com/zk-secrec

Funded by the European Union

Practically, ZK-SecreC models zero-knowledge interactions by enabling a Prover to generate proofs that demonstrate data properties—such as integrity, identity authentication, and regulatory compliance—without revealing any underlying sensitive information. It compiles these high-level proofs into portable execution units that can run on a wide range of platforms, allowing both Provers and Verifiers to execute ZKP programs reliably across different devices and environments. This direct, cross-platform capability ensures that zero-knowledge protocols can be deployed in different TEADAL contexts and operations, providing a robust and practical tool for secure, verifiable data exchanges within the federated ecosystem.

ZK-SecreC enforces trustworthy and verifiable programmability through a set of concrete language features. Its advanced information flow type system ensures strict separation between secret and public data, so that sensitive inputs remain isolated throughout the computation. The language offers built-in functions for operations like fixed-point arithmetic, efficient field-based hashing (using optimized algorithms such as Poseidon), and comparison operations, all of which aid in reducing circuit complexity and ensuring predictable performance. When compiled, ZK-SecreC generates intermediate representations such as SIEVE IR that are compatible with multiple zero-knowledge proof (ZKP) backends. These standardized representations allow integration with established backends like emp-zk[4] and Diet Mac'n'Cheese[5] via Rust interfaces, enabling both interactive and non-interactive proof protocols.

Once complex proof statements are expressed in ZK-SecreC, the compiler produces two distinct executable components: a Prover and a Verifier. The Prover component takes the high-level description and transforms it into arithmetic circuits, subsequently generating ZK proofs. These proofs can follow an interactive protocol — where the Prover and Verifier exchange challenge–response messages in real time — or be converted into non-interactive proofs via techniques such as the Fiat-Shamir heuristic, eliminating the need for back-and-forth communication. The Verifier component, on the other hand, independently validates these proofs, ensuring their correctness without exposing any underlying sensitive data. This separation of roles enables the deployment of Prover and Verifier components in distributed, cross-platform settings. For example, within TEADAL, these components have been integrated into the observability toolchain, where the Prover component intercepts telemetry data and generate verifiable proofs of adherence to service-level agreements for some generic Verifier in a privacy-preserving manner, thereby reinforcing the overall trust framework of the federated ecosystem.

### viii. Synthetic Data Generation

Synthetic Data Generation is emerging as an additional privacy and trust-enhancing mechanism within TEADAL. In early TEADAL iterations, synthetic data was generated to support pilot cases such as evidence-based medicine, regional sustainability, and financial governance. By producing datasets that mimic real-world statistical properties without exposing sensitive information, these early implementations not only protected the privacy of pilots' data but also helped to advance system validation, as well as enhance trust among data providers and consumers.

Building on these initial successes, we evolved our approach to develop an integrated mechanism—the Data Synthesis Workflow Engine (DSWE). Within the TEADAL ecosystem, DSWE has been designed to integrate with existing data ingestion, generation, and sharing components. It serves now as an enabler, allowing synthetic data to become an integral part of the FDP ecosystem. This integration ensures that synthetic outputs can be directly utilized

[4] https://github.com/emp-toolkit/emp-zk
[5] https://github.com/GaloisInc/swanky

Funded by the European Union

alongside real datasets, thereby enhancing overall data trustworthiness and facilitating secure cross-organizational collaboration.

Our DSWE features a modular, pipeline-driven architecture that aligns with TEADAL's comprehensive approach to data management. Core modules include an orchestrator for scheduling and managing synthesis tasks, advanced generative models for realistic data production, and evaluation components that continuously assess both utility and privacy. This design not only ensures flexibility and scalability but also facilitates integration with TEADAL's federated data product mechanisms, enabling synthetic data to be part of a larger, interoperable data ecosystem. Technologically, the engine leverages containerized microservices managed via Kubernetes and orchestrated by workflow platforms such as Kubeflow Pipelines and Argo Workflows. This infrastructure supports reproducible deployments across heterogeneous environments—from local data lakes to cloud-based resources—ensuring that TEADAL's synthetic data processes are both scalable and consistent. Tools like Nix further guarantee that the synthetic data workflows remain reproducible, declarative, and easily auditable.

Security and privacy are also part of the DSWE's design. We designed its architecture to integrate directly with TEADAL's suite of other privacy-enhancing technologies, including Trusted Execution Environments (TEEs) and Secure Multi-Party Computation frameworks. The goal is to be able to turn these synthetic data generation processes into privacy-preserving computations, especially when real data is used for generation. We are able to integrate the same privacy-preserving data pipeline mechanisms in order to offload (to TEEs) or distribute synthetic data generation (MPC, see later) for private execution. This way, the DSWE ensures that generated synthetic data is secure and compliant, thus supporting TEADAL's objective of trustworthy data sharing. Our DSWE not only generates high-quality synthetic data but also enriches the overall federated data ecosystem, ensuring that all data handling processes adhere to key EU regulations targeted by TEADAL—such as the GDPR and the Data Governance Act—thereby reinforcing both data protection and trust over data operations.

The DSWE can integrate into both the Data Ingestion and Federated Data Product (FDP) Pipeline mechanisms. It can be triggered during consumer-provider interactions, dynamically generating synthetic data tailored to various use cases. The engine supports multiple data synthesis types, including Realistic Synthetic Data (RSD) that preserves statistical fidelity, Causal Synthetic Data (CSD) that maintains underlying causal relationships, Artificial Synthetic Data (ASD) based on aggregate rules, and Hybrid Synthetic Data (HSD) which blends these approaches. This flexibility ensures that the synthetic data generated meets the specific requirements of diverse pilot cases and operational scenarios within TEADAL.

Moreover, when integrated with the broader TEADAL Trust Plane, the DSWE enables automatic embedding of utility and privacy assessments directly within the pipeline execution. These embedded evaluations not only ensure that every synthetic dataset is benchmarked for both performance and privacy compliance but also enhance overall auditability and explainability. Similar to other components, these assessments provide data owners with transparent insights into the synthetic data's quality and security. This integration supports compliance with EU regulations, reinforces data sovereignty, and gives data owners means to manage and share synthetic datasets with more confidence. This way, data owners can benefit from transparent, objective assessments that support regulatory compliance and maintain control over data usage in a scalable, reproducible manner.

## ix.TrustOps

Funded by the European Union

TrustOps [5] is a set of practices and a methodology, developed within TEADAL, designed to embed evidence-driven trust throughout the software life cycle. It establishes a continuous cycle of evidence collection, authentication, attestation, and action, ensuring that every development and operational step, from code commits to artifact release, is backed by independently verifiable, tamper-proof records.

At the core of TrustOps lies its evidence life cycle. The process begins with the collection of raw evidence, such as commit logs, build outputs, test results, and runtime telemetry. This raw evidence is then authenticated using cryptographic methods like digital signatures and secure identity verification to confirm its origin and integrity. Once authenticated, the evidence is attested through additional checks, such as remote attestation in TEEs, which verify that the processes and environments conform to established security policies. Finally, the attested evidence is actioned, meaning it is used to authorize subsequent operations, trigger automated responses, or be recorded on immutable ledgers like blockchains for auditability. This cyclical, continuous evidence chain transforms trust from a subjective assumption into an objective, verifiable property.

TEADAL stands to benefit from TrustOps by integrating these evidence-driven practices into its data exchange, federated service operations, and the development of its own software components. By continuously collecting and verifying evidence across its workflows, TEADAL ensures that every interaction from sensitive data ingestion to final data product delivery is fully traceable and compliant with security and regulatory requirements. This approach extends to the entire lifecycle of TEADAL software components, where every phase is underpinned by cryptographic attestations and verifiable evidence. As a result, the entire ecosystem becomes transparent and auditable, reducing manual verification overhead while reinforcing trust, compliance, and operational resilience across the federated data lake architecture.

TEADAL has employed a DevOps methodology for its development and operational practices, enabling rapid iteration and continuous delivery. However, traditional DevOps pipelines often rely on manual verification and lack cryptographic guarantees that every step is executed as intended. TrustOps enhances these practices by embedding an evidence-driven framework into the entire software life cycle. By automating the collection, authentication, and attestation of evidence, from code commits and build outputs to runtime telemetry, TrustOps transforms the DevOps approach into a continuously verifiable process, reducing manual overhead while ensuring robust security and compliance.

However, lack of runtime reproducibility and comprehensive tracking of software artifacts impedes the TrustOps component's capability to accurately evaluate and certify system security, as well as to deploy software that was initially certified as secure. To examine these matters, it is advantageous to introduce the concept of a system dependency graph.

Consider a software runtime environment (e.g., a Linux machine) and all elements employed in constructing its components (compilers, source code, etc.) as a directed multigraph where nodes represent files (executables, libraries, configurations, etc.). A directed edge $x \rightarrow y$ denotes that x is dependent on y, signifying that x was generated from y (e.g., y is the source code from which x was compiled, y is the compiler used to compile x, x is a package containing source code y, etc.) or x necessitates y for operation (e.g., y is a configuration file, a shared library required by x, etc.). This graph shall be termed the system dependency graph. It is well-established that this graph undergoes frequent modification over time for a given system (e.g., source code revision, security update, etc.), thus the graph from a previous day may differ from the present one.

Tracking the evolution of this graph over time presents significant challenges when employing conventional engineering tools and methodologies. While nodes are frequently

Funded by the European Union

associated with version numbers, these function merely as symbolic labels assigned by individuals to software artifacts, and their uniqueness as identifiers cannot be guaranteed. Instances such as an attacker compromising a library or a developer inadvertently omitting a version number update following minor library module modifications serve as illustrations. (While Continuous Integration/Continuous Deployment practices can mitigate this issue, they do not offer a comprehensive resolution.) Further complicating the matter is the potential for developers to incompletely document the dependencies of their source code or the precise versions of the software tools utilized during the build process.

Security assessment and certification processes necessitate reference to a specific system dependency graph. This requirement arises from the temporal nature of security assessments, which can only evaluate system security at a particular point in time, contingent upon the system dependency graph existing at that juncture. However, identifying the precise graph operational during the assessment can prove challenging. Consequently, security assessments commonly rely on version numbers, which may hinder the definitive determination of system security, even when components share identical version numbers with those evaluated in the assessment, thus complicating the applicability of the assessment to the system under consideration.

This matter is connected to the challenge of replicating the system dependency graph utilized during the certification process, which is considerably complex when employing conventional package management and deployment tools. Consequently, even after system construction, a definitive assurance that the security evaluation conducted during certification remains valid for the current system cannot be provided. As a specific scenario, the challenge of reinstating a system to a known secure state after a security incident—specifically, the state the system was in during a successful security evaluation—is notable. Executing this action proves difficult without precise knowledge of the system dependency graph as it existed during the evaluation. Furthermore, in practical application, the restoration procedure can be cumbersome and protracted, despite the application of an “Infrastructure as Code” methodology.

To overcome these challenges, the TrustOps component adopts reproducible builds based on complete dependency graphs. The implementation relies on Nix, a purely functional programming language and package manager which offers a solution to precisely and unambiguously describe a system dependency graph. This capability allows the TrustOps component to reference specific analyzed graphs, enables the reproduction of certified secure systems, and facilitates easy rollback to known secure states. Moreover, the deployed system can be monitored to detect whether the current dependency graph differs from the one which the TrustOps component assessed. A difference in the graph could be a telltale sign of a security breach. In any case, the Teadal Node Operator can easily roll back the system to the state that the TrustOps component assessed.

A concrete example of TrustOps integration in TEADAL is our new architecture for Trustworthy CI pipelines, where every component of the pipeline interacts to form an end-to-end, evidence-driven chain. The process starts with an authenticated version control system that enforces digital signatures on every commit, ensuring source code integrity and developer accountability. A workflow orchestration engine (using platforms like Argo Workflows or GitLab pipelines) manages task scheduling and dependency resolution, ensuring that every step adheres to predefined security policies. Sensitive tasks are executed within Trusted Execution Environments (TEEs), where a deterministic build system such as Nix guarantees that identical inputs always produce identical outputs in a secure, isolated context. Complementing these components, secure key management and remote attestation provide cryptographic proof that the TEE environment is genuine, while blockchain commitments record critical cryptographic hashes and signatures of build and test artifacts, creating an immutable, tamper-proof audit trail.

Funded by the European Union

These interconnected components collectively deliver strong security guarantees throughout the CI process. The authenticated version control system assures that every code change is verifiable, while the orchestration engine ensures coordinated and secure execution of tasks. The TEE-based execution environment, with its embedded deterministic build system, minimizes the risk of tampering and inconsistencies, and any deviation is immediately detected. Secure key management and remote attestation further help the trustworthiness of the process by confirming the integrity of the execution environment. Finally, the artifact registry, integrated with blockchain commitments, stores signed build, test, and audit outputs in an immutable ledger that can be independently verified. This framework results in attestable artifacts that provide concrete, cryptographic evidence of compliance and integrity. Ultimately, by integrating blockchain-anchored, attestable artifacts into TEADAL's CI pipelines, every individual software component—whether it's the data ingestion modules, federated data product generators, identity management systems, or observability tools—gains a verifiable evidence trail that guarantees build integrity, minimizes independent verification overhead, and enables secure, compliant interactions across the entire TEADAL ecosystem.

## 1. Further Efforts

Building on the security assurances established by the TrustOps methodology, additional work is underway to streamline and automate the deployment and operation of TEADAL nodes across diverse environments. These enhancements aim to make it easier to adopt best practices and maintain consistent security postures without requiring specialized infrastructure expertise.

A standardized Terraform module is being developed to bootstrap and manage Kubernetes-based infrastructure in a fully automated manner. The module abstracts complex infrastructure logic and provides a simplified interface that requires only Terraform usage for deployment and management. With minimal initial input from a system administrator, it can provision a fully functional, production-ready Kubernetes cluster. The design ensures that no deep understanding of the underlying infrastructure is necessary, streamlining the setup process and enabling fast, consistent adoption across various environments.

The module is meant to be cloud-agnostic and supports multiple platforms, including OpenNebula, OpenStack, VMware, and Proxmox. It utilizes Talos Linux as the operating system for Kubernetes nodes, eliminating the need for traditional configuration management and increasing overall security and consistency. Internal Kubernetes components can be defined, enabled, or disabled based on project-specific requirements. Built-in tools are included for monitoring both the cluster and the workloads deployed into it.

To promote secure-by-default deployments, the module enforces a set of best security practices where technically feasible. These enforced practices are governed by established patterns to ensure consistency and compliance across environments. For CI/CD workflows, Argo CD is integrated for continuous delivery, while Argo Workflows handles the continuous integration side, providing a GitOps-native, declarative and automated pipeline experience. This approach eventually helps us form a reliable baseline for standardizing Kubernetes infrastructure and operations across teams and projects.

By unifying the provisioning of Kubernetes clusters, enforcing security best practices, and supporting GitOps-based continuous delivery, we plan to make this module extend the TrustOps vision to infrastructure management. The plan is to ensure that every step from code commit through runtime deployment is cryptographically verifiable and automatically tracked, reducing the operational overhead and risks associated with deploying TEADAL nodes. Organizations then can benefit from shortened setup times, reliable rollouts, consistent configuration baselines, and end-to-end traceability of changes. In turn, these

Funded by the European Union

improvements can help the efficiency, reliability, and security of TEADAL environments, helping to meet compliance needs while maintaining high-performance data flows and rapid innovation cycles.

Funded by the European Union

# 3 Demonstration and Evaluations

In this section, we apply the mechanisms and trustworthy infrastructure components on the use cases of TEADAL on two scenarios to prove the capabilities of the TEADAL Trust Plane.

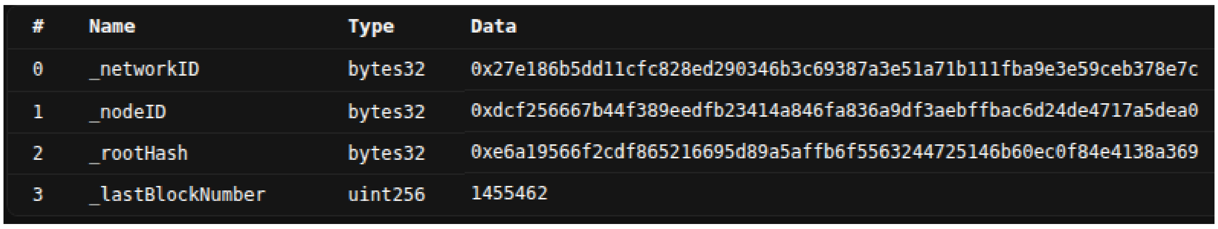
The TEADAL baseline includes all TEADAL tools and serves as the foundation for further customization. The TEADAL node is deployed on a Kubernetes cluster. It includes general-purpose tools such as Istio, ArgoCD, Minio or Jaeger, as well as TEADAL-specific components, such as the Catalogue, Advocate or a Policy Manager. As described in deliverables D6.1 and D6.2 [10, 11], TEADAL GitLab hosts a central TEADAL node repository at https://gitlab.teadal.ubiwhere.com/teadal-tech/teadal.node. The TEADAL node is installed and running on all use-case specific scenarios.

## B. Blockchain-based trust mechanisms – Identity Trust

**Blockchain State Anchoring** was successfully tested on a development environment and the private blockchain used for testing purposes was anchored to the public Sepolia Ethereum testnet.

The following is one of the transactions submitted to the Sepolia testnet:

- 0x3504160449c39b81bb548f0b771230fae6ac3de8f94817d8092757c0fa16ca5e



| # | Name | Type | Data |
|---|---|---|---|
| 0 | _networkID | bytes32 | 0x27e186b5dd11cfc828ed290346b3c69387a3e51a71b111fba9e3e59ceb378e7c |
| 1 | _nodeID | bytes32 | 0xdcf256667b44f389eedfb23414a846fa836a9df3aebffbac6d24de4717a5dea0 |
| 2 | _rootHash | bytes32 | 0xe6a19566f2cdf865216695d89a5affb6f5563244725146b60ec0f84e4138a369 |
| 3 | _lastBlockNumber | uint256 | 1455462 |

*Figure 11: Inputs of the example transaction.*

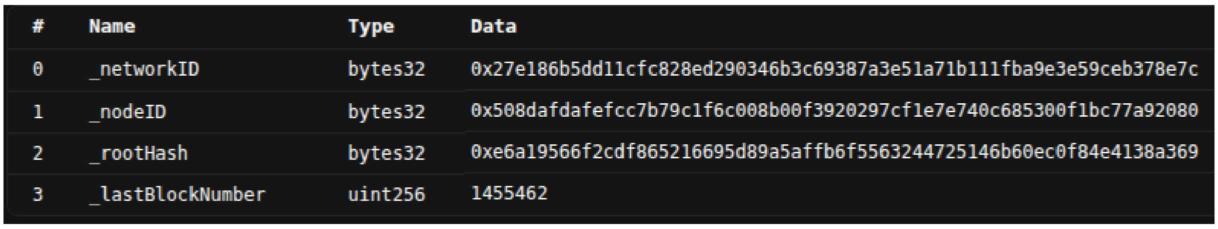
Where we can see the input parameters as:

- _networkId:
  - **0x27e186b5dd11cfc828ed290346b3c69387a3e51a71b111fba9e3e59ceb378e7c:** The Network identifier, as the resulting sha256 digest of the "Teadal-Blockchain" name.
- _nodeId:
  - **0xdcf256667b44f389eedfb23414a846fa836a9df3aebffbac6d24de4717a5dea0:** The node identifier, as the resulting sha256 digest of the "teadal-validator-3" name.
- _rootHash:
  - **0xe6a19566f2cdf865216695d89a5affb6f5563244725146b60ec0f84e4138a369:** The hash of the block that is anchored with the current transaction
- _lastBlockNumber:
  - **1455462:** The current block number that is being anchored

To validate the state, another transaction by a different node can be red:

- 0x6b1874cebf6467f98811e2bde0dd013d71541791aa777e584a0c7c05153e24aa

Funded by the European Union



| # | Name | Type | Data |
|---|---|---|---|
| 0 | _networkID | bytes32 | 0x27e186b5dd11cfc828ed290346b3c69387a3e51a71b111fba9e3e59ceb378e7c |
| 1 | _nodeID | bytes32 | 0x508dafdafefcc7b79c1f6c008b00f3920297cf1e7e740c685300f1bc77a92080 |
| 2 | _rootHash | bytes32 | 0xe6a19566f2cdf865216695d89a5affb6f5563244725146b60ec0f84e4138a369 |
| 3 | _lastBlockNumber | uint256 | 1455462 |

*FIGURE 12: INPUTS OF THE EXAMPLE TRANSACTION BY A DIFFERENT NODE.*

We can see that the transaction is about the same network (same _networkId) but from a different node.

The two nodes are correctly synchronized and they agreed on the state of the Teadal-Blockchain network at block number 1455462.

**Passkeys** mark a significant advancement over traditional password-based authentication systems. Below are the main advantages that make this technology particularly secure and versatile:

- Password Elimination: Passkeys eliminate the risks associated with stolen or compromised passwords.
- Robust Cryptography: The secp256K1 curve ensures a high level of security.
- Private Key Protection: The private key is stored locally on the user's device, ensuring it is never exposed during authentication. Tools like MetaMask serve as wallets to securely store keys and simplify access to signing features.
- Updates and Revocation: If the private key is compromised, the associated DID can be updated to replace the compromised key with a new one, ensuring continuity of identity and security.
- Interoperability: Leveraging the DID standard, passkeys can be used across multiple platforms, enabling secure, portable, and decentralized identity management.

The inclusion of the secp256k1 curve in the official COSE registry, as outlined in RFC 8812 – "Registrations for WebAuthn Algorithms in COSE and JOSE", formally establishes support for the ES256K algorithm and the secp256k1 curve within the COSE signature framework one of the foundational components of the WebAuthn protocol. For reference, the official document can be found here: https://www.rfc-editor.org/rfc/rfc8812.

That said, it's worth noting that despite this standardization, practical support for secp256k1 across FIDO2 devices and mainstream browsers remains limited. Popular implementations—such as YubiKey, Google Titan, and SoloKeys—as well as WebAuthn stacks in browsers like Chrome, Firefox, and Safari continue to favor the secp256r1 curve. This choice aligns with current NIST recommendations and the capabilities of widely adopted cryptographic hardware.

As a result, using secp256k1-based passkeys in blockchain-related environments currently requires alternative tooling—for example, cryptographic wallets like MetaMask or Ledger, and decentralized identity protocols (DIDs) as previously discussed. In Web3 contexts, this approach offers a strategic advantage: it allows identity, authentication, and digital signatures to converge on a single key, minimizing redundancy and streamlining interoperability between conventional systems and blockchain-based infrastructure.

**Smart Contract for DIDs** offer an advanced infrastructure for managing decentralized identities, ensuring transparency, immutability, and security. The addition of advanced features such as multisig wallet support and identity ownership tracking further enhances the trust and security of the system, allowing multiple entities to participate in the

Funded by the European Union

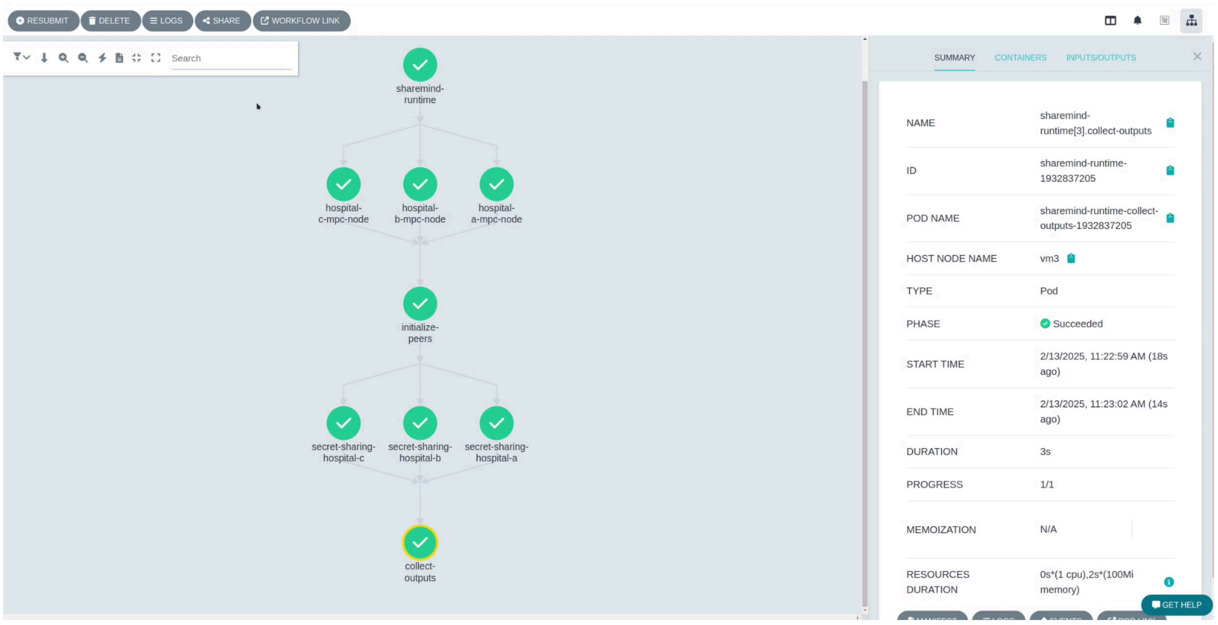
decision-making process. This approach is fundamental to building identity systems that respect the principles of digital sovereignty and distributed trust.

There isn't a single "official" smart contract for DIDs on a blockchain. The W3C only lays out the method specs (the DID Core rules and the Method Registry) and leaves it up to each method to build its own implementation . One of the earliest and most widely adopted solutions is uPort's Ethr-DID-Registry (ERC-1056)

Available at: https://github.com/uport-project/ethr-did-registry

## C. Privacy-preserving computation components - Computational Trust

**Sharemind MPC tooling and technology** has been developed throughout the project. We have successfully demonstratd Sharemind MPC runtime integration with TEADAL baseline technologies. We demonstrated how to encode privacy-preserving computations in the SecreC programming language and ran the demonstrator with three computing nodes, generating a large output dataset from three private input datasets. The setup mimics the evidence-based medicine pilot scenario (see Figure 9). Next iterations will focus on gathering benchmarks and performing technical evaluation.

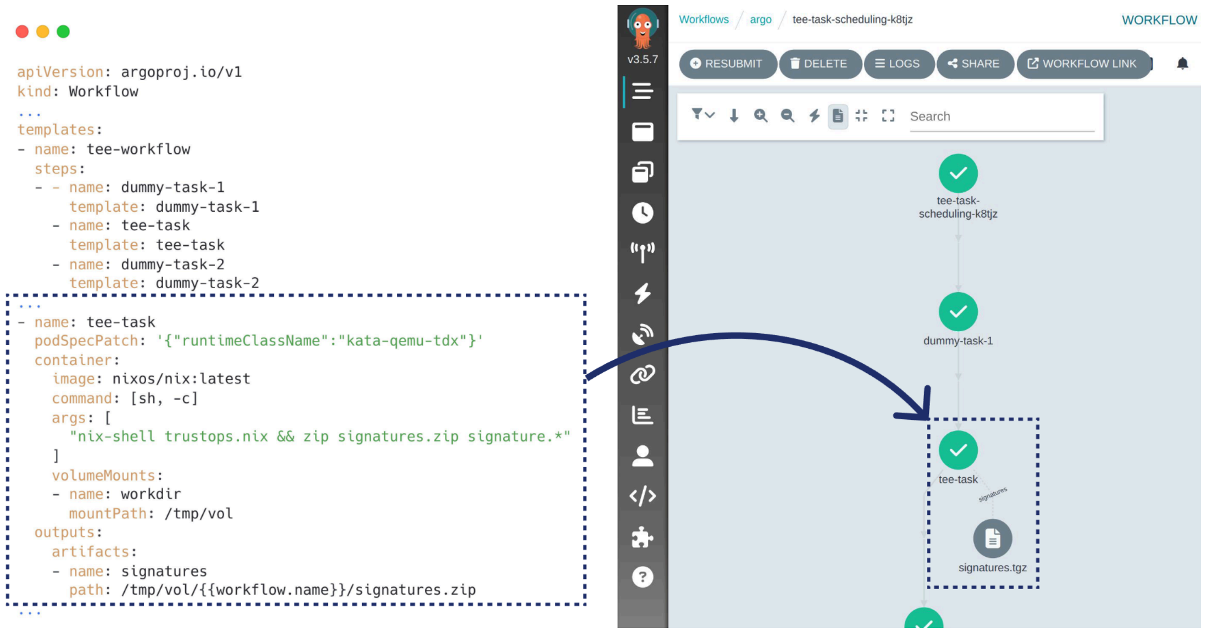


| | |
|---|---|
| NAME | sharemind-runtime[3].collect-outputs |
| ID | sharemind-runtime-1932837205 |
| POD NAME | sharemind-runtime-collect-outputs-1932837205 |
| HOST NODE NAME | vm3 |
| TYPE | Pod |
| PHASE | ✓ Succeeded |
| START TIME | 2/13/2025, 11:22:59 AM (18s ago) |
| END TIME | 2/13/2025, 11:23:02 AM (14s ago) |
| DURATION | 3s |
| PROGRESS | 1/1 |
| MEMOIZATION | N/A |
| RESOURCES DURATION | 0s*(1 cpu),2s*(100Mi memory) |

*Figure 13: Sharemind MPC distributed pipeline demonstrator for the evidence-based medicine pilot.*

**TEE-based pipelines** were successfully demonstrated in D5.2 [2], inspired by the former shared financial governance use case (see Figure 10). We demonstrated the programmability, integration, and execution of privacy-preserving data pipelines, in a distributed setting consisting of a multi-node Kubernetes cluster, where private tasks in a pipeline are offloaded and orchestrated to run on trusted confidential computing units, even if the host system is untrusted. Previous deliverable D4.2 [12] and parallel deliverable D3.3, further detail the implementation, assessment, and validation procedures.

Funded by the European Union



```yaml
apiVersion: argoproj.io/v1
kind: Workflow
...
templates:
- name: tee-workflow
  steps:
  - - name: dummy-task-1
      template: dummy-task-1
    - name: tee-task
      template: tee-task
    - name: dummy-task-2
      template: dummy-task-2
...
- name: tee-task
  podSpecPatch: '{"runtimeClassName":"kata-qemu-tdx"}'
  container:
    image: nixos/nix:latest
    command: [sh, -c]
    args: [
      "nix-shell trustops.nix && zip signatures.zip signature.*"
    ]
    volumeMounts:
    - name: workdir
      mountPath: /tmp/vol
  outputs:
    artifacts:
    - name: signatures
      path: /tmp/vol/{{workflow.name}}/signatures.zip
...
```

FIGURE 14: ARGO WORKFLOW EXAMPLE PIPELINE, AS YAML FILE ON THE LEFT, COMPRISING ONE TEE-BASED TASK, LABELED VIA THE RUNTIMECLASSNAME ATTRIBUTE AS RUNNING INSIDE AN INTEL TDX VM, AND ITS EXECUTION AND VISUALIZATION ON THE RIGHT.

We have been augmenting and developing **ZK-SecreC**[6] [7], enriching its set of standard library functionalities and program demonstrators. The code is now open source and we have been maintaining it. We have successfully encoded verifiability statements in ZK-SecreC, specifically for SLA verification, detailed in D3.3. We created an initial prototype and demonstrated its integration with monitoring and observability tooling used in TEADAL, as seen in Figure 11. Tables 1 and 2 also show comparisons of benchmarks between a ZK-SecreC implementation and a ZK-SNARK (gnark[8]) implementation.

In evaluating interactive (ZK-SecreC) and non-interactive (zk-SNARK) ZKP pipelines for continuous SLA monitoring, both implementations showed distinct trade-offs in performance and architectural complexity (see Figures 12 and 13). The zk-SNARK pipeline exhibited superior runtime and resource metrics—achieving verification speeds nearly 10× faster and using significantly less CPU and memory on the verifier side—thanks to its asynchronous, decoupled design and succinct proofs. However, these gains come at the cost of requiring a trusted setup and higher cryptographic development overhead. In contrast, the ZK-SecreC-based interactive approach, while slower and more tightly coupled due to its synchronous design and reliance on active network communication, offered considerable advantages in developer ergonomics and programmability. Its modular Rust-like syntax and direct circuit expressiveness made it more accessible for defining complex SLA verification logic. Thus, while zk-SNARKs are better suited for scalable, fault-tolerant deployments, ZK-SecreC remains a valuable tool for prototyping and implementing advanced, logic-intensive verification tasks where rapid iteration and rich expressiveness are key.

[6] https://zk-secrec.cyber.ee/documentation/
[7] https://github.com/zk-secrec
[8] https://github.com/Consensys/gnark

Funded by the European Union

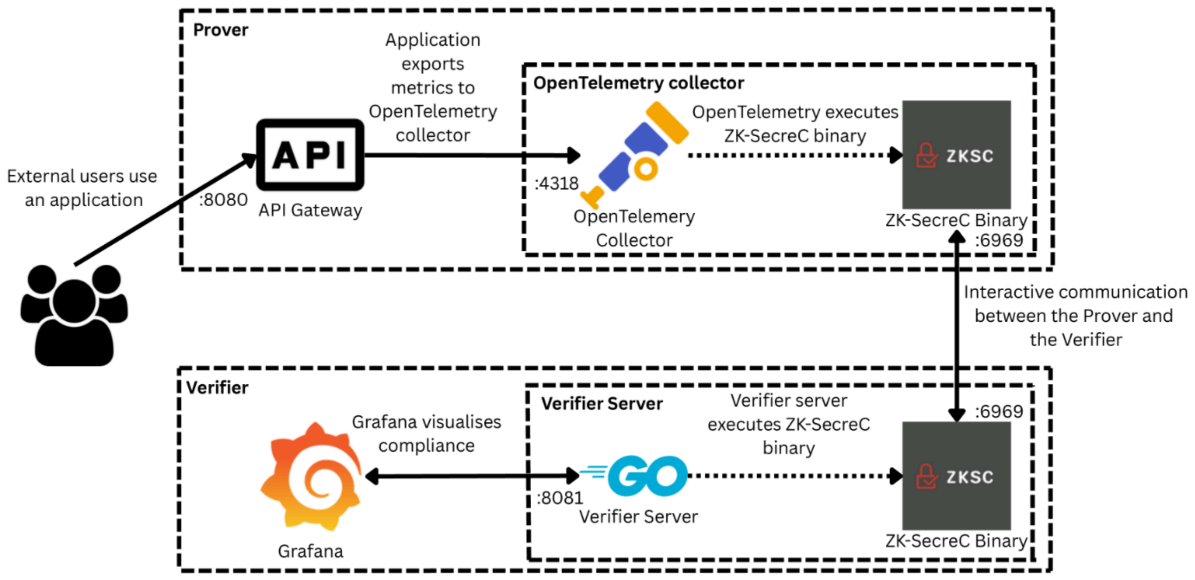

Prover
Application exports metrics to OpenTelemetry collector
OpenTelemetry collector
OpenTelemetry executes ZK-SecreC binary
External users use an application
API
:8080
API Gateway
:4318
OpenTelemery Collector
ZKSC
ZK-SecreC Binary
:6969
Interactive communication between the Prover and the Verifier
Verifier
Verifier Server
Verifier server executes ZK-SecreC binary
Grafana visualises compliance
:6969
GO
:8081
Verifier Server
ZKSC
Grafana
ZK-SecreC Binary

*FIGURE 15: ARCHITECTURE OF THE PROTOTYPE INTEGRATION OF ZK-SECREC WITH OPENTELEMETRY TOOLING FOR SLA VERIFIABILITY.*

| Metric | ZK-SecreC | zk-SNARK |
|---|---|---|
| Prover Avg. Execution Time | 3432.12 ms | 306.48 ms |
| Prover Min/Max Execution Time | 3073.00 / 3673.00 ms | 252.00 / 367.00 ms |
| Verifier Avg. Execution Time | 3332.17 ms | 5.30 ms |
| Verifier Min/Max Execution Time | 2978.00 / 3553.00 ms | 3.00 / 11.00 ms |

*TABLE 1: EXECUTION TIME COMPARISON BETWEEN ZK-SECREC AND ZK-SNARK IMPLEMENTATIONS.*

| Metric | ZK-SecreC | zk-SNARK |
|---|---|---|
| Prover Avg. CPU usage | 33.11% | 7.13% |
| Verifier Avg. CPU usage | 29.68% | 1.02% |
| Prover Avg. Memory usage | 1231.55 MB | 1608.27 MB |
| Verifier Avg. Memory usage | 1545.53 MB | 962.24 MB |

*TABLE 2: RESOURCE USAGE COMPARISON BETWEEN ZK-SECREC AND ZK-SNARK IMPLEMENTATIONS.*

For **zkML**, we studied the Easy Zero-Knowledge Inference (EZKL) library, tested its functionality, and applied it to a real-world use case. Our work resulted in a paper, which was presented as a demo paper at the Brains 2024 conference in Berlin [4].

## D. TRUST MANAGEMENT TOOLS - BEHAVIOR TRUST

**Advocate** is installed and running in the three use-case specific scenarios evidence-based medicine, smart viticulture and industry 4.0. Additionally, Advocate is installed on two use-case independent scenarios, showcasing its usage for energy certification and in a distributed market. To gain insights into the practicality of Advocate, we execute a set of experimental test cases, measuring the performance of the claim creation. These tests assess the following operations: building (build) the claim, hashing (keccak) and signing (sign) the claim, storing the claim in IPFS (ipfs), hashing (sha256) and anchoring it on-chain (blockchain_tx), and storing it in a database (db_tx). The results of those measurements are depicted below.

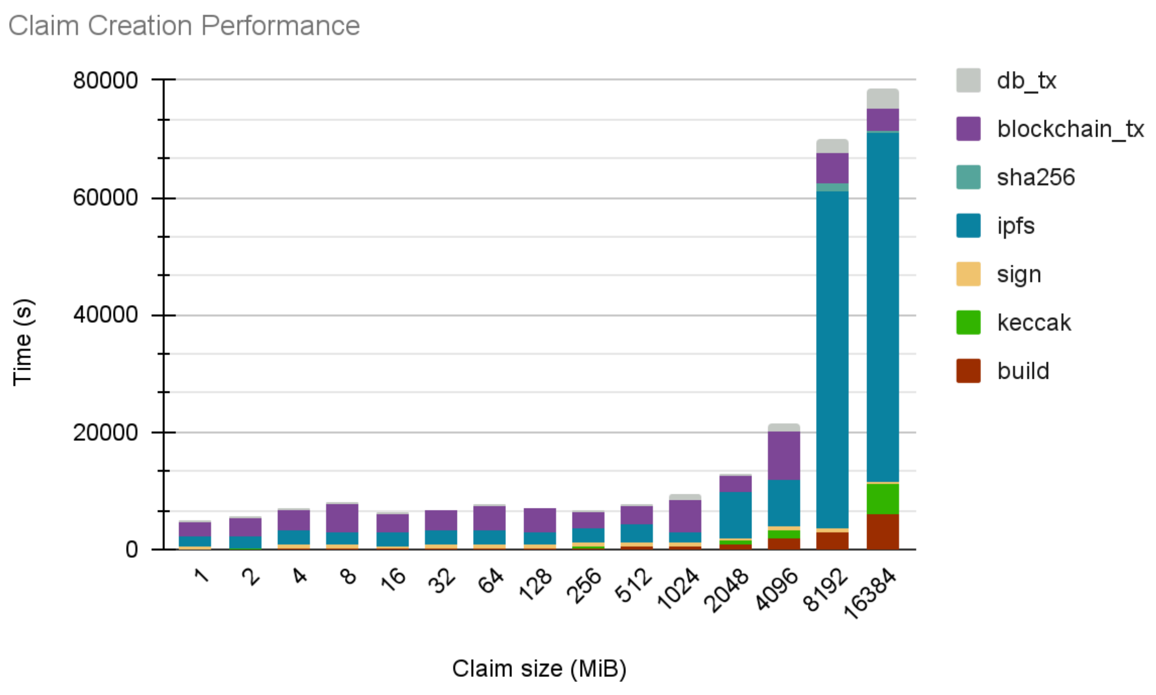


*FIGURE 16: : CLAIM CREATION PERFORMANCE.*

The measurements indicate that with an increasing claim size, the total time required for claim creation also increases. For smaller sizes, up to 1024 MiB, the total time increases gradually. However, beyond 2048 MiB, there is a noticeable rise in the total processing time. The IPFS becomes the most performance-intensive part of the claim creation for larger claims. From 4096 MiB, it exceeds the remaining operations' performance. This suggests that initiating the storage in IPFS may represent a bottleneck in the claim creation for larger claims. In contrast, operations such as signing, hashing with Keccak and SHA-256, and even blockchain and database transactions contribute relatively little to the total performance of the claim creation.

For the **TNS/Federation Smart Contract**, we provide measurements of the gas costs associated with deployment and execution within the TEADAL ecosystem. The gas cost directly impacts the economic feasibility of interacting with Smart Contracts. Higher gas costs translate to higher transaction fees for users.

Table 1 focuses on the gas costs associated with deploying the three smart contracts: Federation, DocumentStore, and ENSRegistry. Deployment gas costs are incurred only once when the contract is initially deployed to the blockchain.

| Smart Contract | Federation | Document Store | ENSRegistry |
|---|---|---|---|
| Gas cost | 4,669,569 | 1,796,962 | 1,084,616 |

*TABLE 3: SMART CONTRACT DEPLOYMENT GAS COST.*

Funded by the European Union

Our results show that the Federation contract has the highest deployment cost at 4,669,569 gas. In contrast, the ENSRegistry contract has the lowest deployment cost, at 1,084,616 gas. The DocumentStore contract's deployment cost of 1,796,962 gas falls between the other two.

Table 2 breaks down the minimum, maximum and average gas costs for executing functions within these smart contracts:

| Smart Contract | Function | Gas cost |
|---|---|---|
| **Federation** | *addDataset* | 127,944 |
| | *removeDataset* | 43,794 |
| | *approveMember* | 103,697 |
| | *removeMember* | 36,395 |
| | *registerPolicy* | 100,351 |
| | *deregisterPolicy* | 36,703 |
| | *publishMetadata* | 42,358 |
| | *rateDataset* | 103,314 |
| | *raiseConflict* | 120,755 |
| | *resolveConflict* | 54,267 |
| | *trackDataLineage* | 29,548 |
| **DocumentStore** | *issue* | 48,312 |
| | *revoke* | 48,528 |
| **ENSRegistry** | *setResolver* | 48,837 |
| | *setSubnodeOwner* | 50,240 |

*TABLE 4: SMART CONTRACT FUNCTION EXECUTION GAS COST.*

Our results show that the most gas-intensive functions of the Federation contract are *addDataset* (127,944 gas), *raiseConflict* (120,755 gas), and *approveMember* (103,697 gas). Simpler functions such as *trackDataLineage* (29,548 gas), *removeMember* (36,395 gas), and *deregisterPolicy* (36,703 gas) have lower associated gas costs. The DocumentStore contract shows consistent gas usage for *issue* (48,312 gas) and *revoke* (48,528 gas). The ENSRegistry contract functions *setResolver* (48,837 gas) and *setSubnodeOwner* (50,240 gas) show moderate gas costs.

The **Catalog transaction Observer** feature is implemented in the TEADAL Federated Metadata Catalogue by means of Service Tasks in BPMN processes. At the time of writing,

Funded by the European Union

the logging of Catalogue events to Advocate has been integrated for these processes or user requests:

- asset creation and publication (lifecycle process)
- asset deletion (user request)
- FDP contract creation request (user request)

The implementation of the **Data Synthesis Workflow Engine (DSWE)** demonstrated its technological and integration feasibility in the context of TEADAL tools and architecture. We performed synthetic data generation for pilots in the initial phases of the project, detailed in D2.1, D2.2, and D2.3 [7, 8, 9]. We modelled the synthetic data pipelines and datasets according to the pilot requirements, schemas from real datasets, and other use case goals. We have successfully generated datasets that serve ongoing validation purposes.

To assess scalability, we benchmarked it on use case datasets of 1K, 10K, and 100K rows using a cloud VM (8 vCPUs, 12GB RAM). The pipeline included data loading, preprocessing, generation, and parallel evaluation (quality, diagnostic, privacy). As shown in Figure 15, loading and preprocessing remained constant (around 0.1 min), while generation rose from 0.14 to 5.35 min, privacy from 0.14 to 9.46 min, and quality from 0.10 to 5.59 min. Total runtime scaled sublinearly: 1.6 min (1K), 5.1 min (10K), and 16 min (100K). At 100K rows, privacy and quality evaluations accounted for over 90% of runtime, but with parallel execution via Kubeflow, their durations didn't accumulate. These results confirm the DSWE scalability and show how modular, concurrent design mitigates bottlenecks at scale.

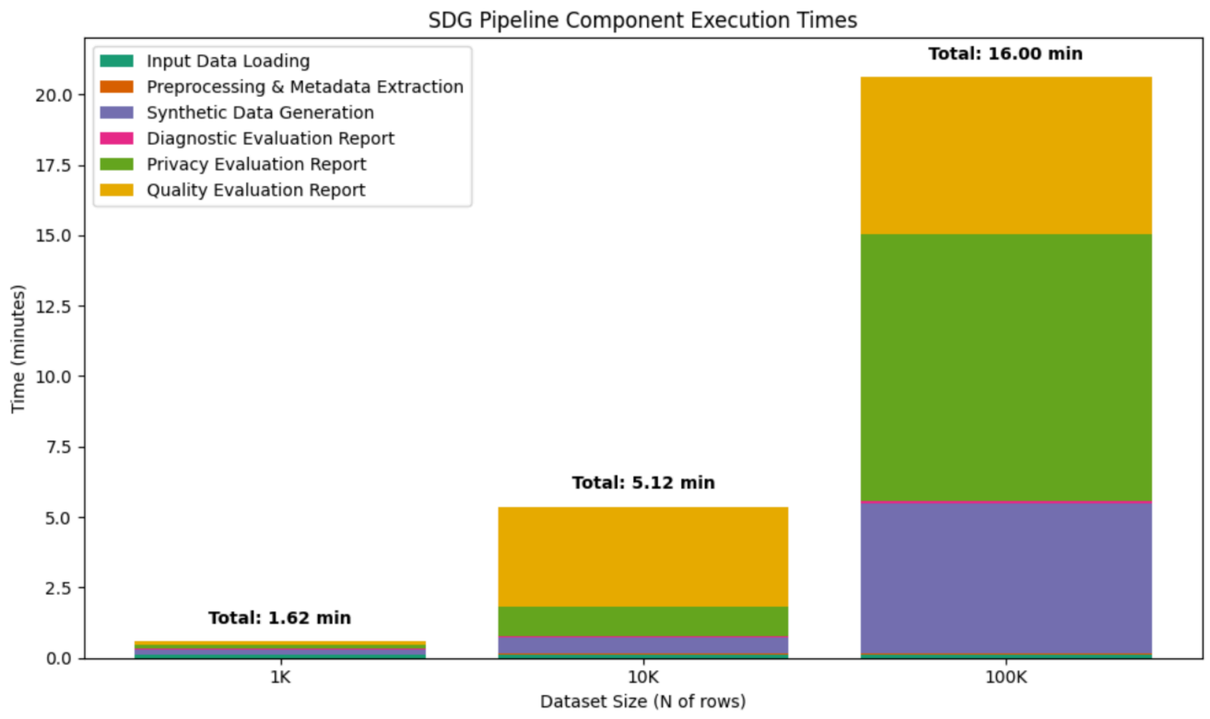


*FIGURE 17: EXECUTION TIME PER SDG PIPELINE COMPONENT ACROSS DATASET SIZES.*

Following **TrustOps** principles, we created a first prototype of the Trustworthy CI pipeline and demonstrated its potential integration with TEADAL-related baseline tooling. We developed a prototypical scenario for generating attestable artifacts from build, test, and audit phases of a piece of software (see Figure 16). We also ran initial benchmarks and established scalability estimations for larger and more frequent continuous integration processes, in Figure 15.

Funded by the European Union

Our evaluation shows that integrating Trusted Execution Environments (TEEs) into CI pipelines introduces moderate, controlled overhead—such as a 3.35-minute increase in workflow time and higher CPU/memory usage per task—but this is offset by significant trust and scalability benefits in software supply chains. Despite lacking the runtime optimizations of mature Docker-based systems, our TEE-based prototype demonstrates practical viability, with the additional costs amortized across multiple Consumers who benefit from lightweight, cryptographically verifiable artifacts (Figure 17). A simulation modeling CI scaling confirms that our approach drastically reduces redundant re-execution efforts as Consumer numbers grow, offering near-constant Producer-side costs (Figure 18). Moreover, by leveraging deterministic builds and secure enclave attestations, our system ensures early detection of tampering or misbehavior, thereby eliminating wasted verification cycles and enhancing artifact trustworthiness. These results affirm the potential of TEE-backed architectures to deliver scalable, efficient, and secure CI pipelines, with future improvements expected through deeper orchestration and virtualization integration.

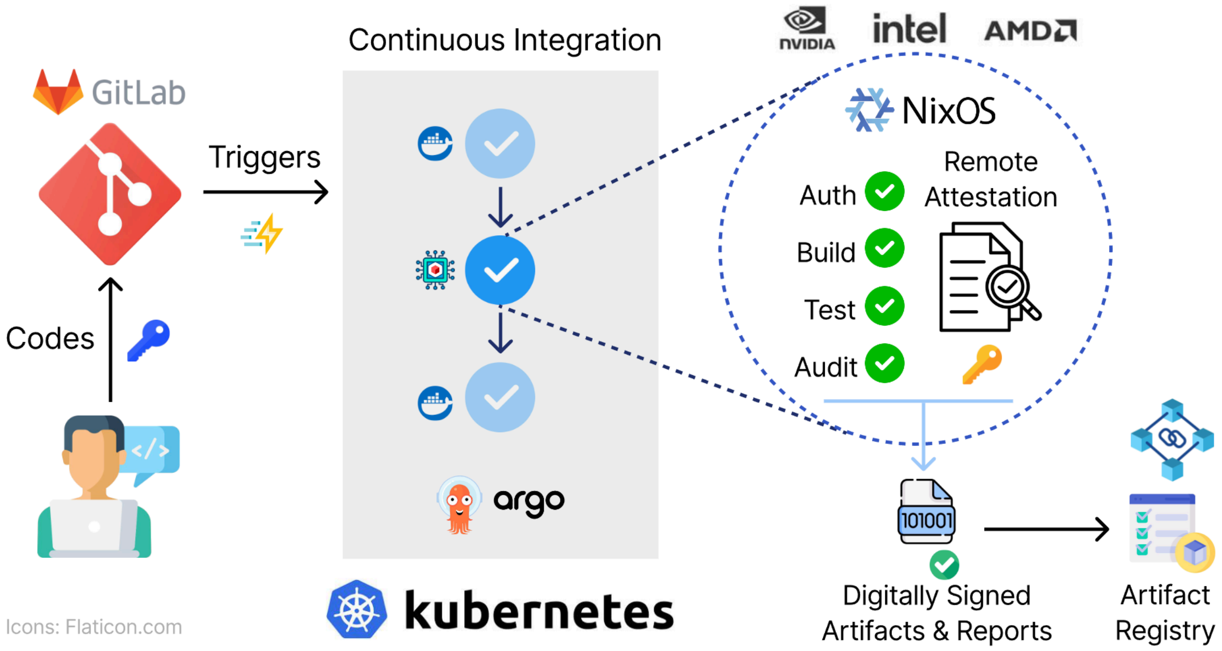


FIGURE 18: EVIDENCE-DRIVEN TRUSTWORTHY CI PIPELINE IMPLEMENTATION.

| Metric | Without | With |
|---|---|---|
| Total Workflow Time (min) | 4.15 | 7.50 |
| CPU Consumption (s per CPU) | 27 | 49 |
| Memory Consumption (s per 100MiB) | 6 | 12 |
| Runtime of Tasks (s) | 65.8 | 141.5 |
| Evidence Size (B per signature) | N/A | 294 |
| Cryptographic Operation Time (ms per Op) | N/A | 80 |
| Blockchain Commit Time (ms) | N/A | 213 |

FIGURE 17: AVERAGE COMPARISON RESULTS OF 10 RUNS.

Funded by the European Union

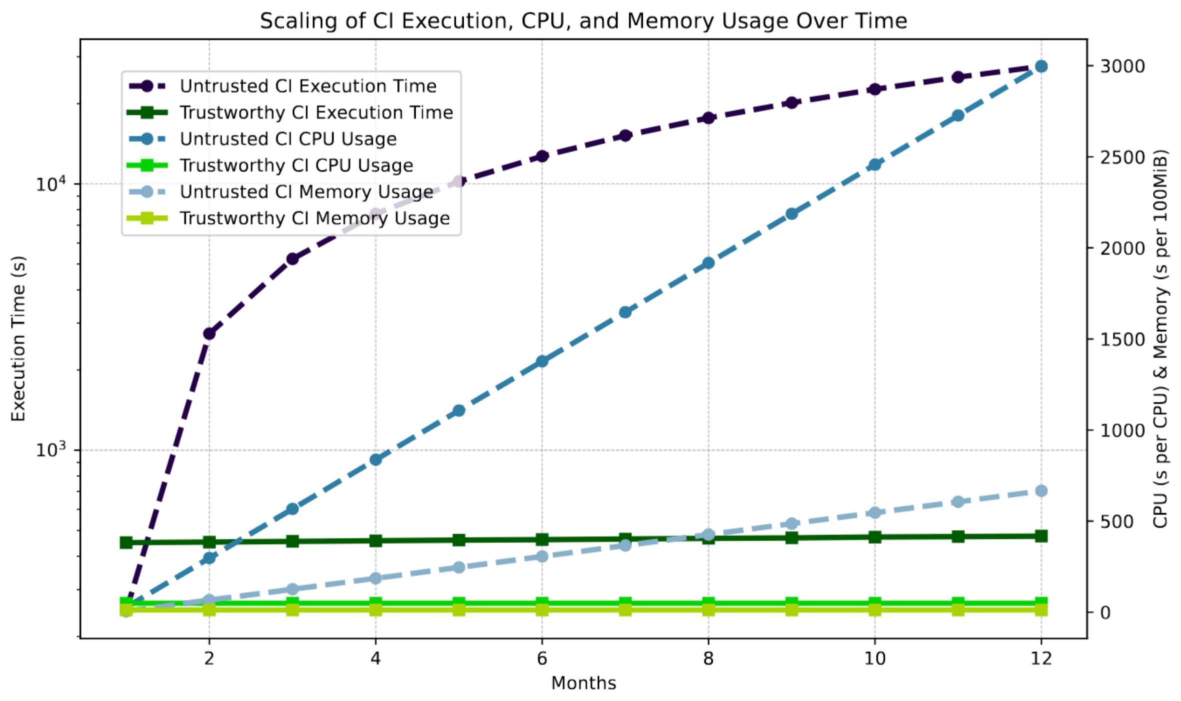


*Figure 19: Scaling of CI execution, CPU, and memory usage over time, for a growth rate of 10 consumers per month and a new release every month.*

## E. Use-case Independent Scenarios

To demonstrate advanced capabilities of the Trust Plane within the TEADAL ecosystem, two use-case independent scenarios have been developed. The following section describes two fundamental scenarios: one focused on energy certification and the other on a distributed market interaction.

### i. Energy Certification

The energy certification scenario represents the first use-case independent scenario developed. It involves a computing resource provider within a federated environment, offering processing capabilities, and an internal service consumer that requires transparency and verifiability of the energy consumed by these resources. A key requirement from the consumer is the ability to independently verify the energy usage associated with their workloads. To support this, Advocate acts as an independent verifier, issuing claims based on selected energy consumption measurements related to the processing activities. These claims are derived from the provider's usage metrics using Kepler. By accessing this data, it is possible to generate signed attestations that confirm energy usage, enabling the consumer to verify whether their resource utilization aligns with expected or target energy consumption levels. This ensures accountability and builds trust within the organization, since the integrity and non-repudiation of energy consumption claims is maintained.

### ii. Distributed Market

The distributed market scenario represents the second use-case independent scenario that was developed to showcase the capabilities of the Trust Plane. It involves a collaboration between a service provider, offering data processing capabilities (e.g., compute resources or storage), and a service consumer, that submits data for processing and expects to be billed

Funded by the European Union

based on actual resource consumption. A key requirement from the consumer is transparency and verifiability of the billed costs. To support this, Advocate acts as an independent verifier, issuing claims based on selected cost measurements related to the processing activities. These claims are derived from the provider's usage metrics using OpenCost. By accessing this data, signed attestations are generated that confirm resource usage and corresponding costs, enabling the consumer to verify whether charges align with actual consumption. This ensures accountability and builds trust between the parties, since integrity and non-repudiation of billing claims is maintained.

## F. Use-case Specific Scenarios

To demonstrate the integration of selected components of the TEADAL Trust Plane, we now revisit their application within the five pilot cases of TEADAL. These pilots include: Evidence-Based Medicine, Mobility, Smart Viticulture, Industry 4.0, and Shared Financial Data Governance. This section focuses on how tools and components have been incorporated into the pilots to support trustworthy data sharing across federated environments.

### i. USE CASE PILOT #1: EVIDENCE-BASED MEDICINE

The evidence-based medicine scenario simulates a clinical study where medical parameters are selected in a web-based client. The system comprises three nodes, two that are hospitals and one that is both hospital and researcher who will run a study. A description of the evidence-based medicine pilot can be found in deliverable D2.1, D2.2 and D2.3 [7, 8, 9] under the "USE CASE PILOT #1: EVIDENCE-BASED MEDICINE" chapter.

**Advocate** collects interaction evidence for the purposes of auditing and establishes a chain of trust between data providers and data consumers. This is important in a medical use case, where sensitive patient data is involved. By closely observing the entire data-sharing lifecycle, Advocate ensures that all interactions adhere to predefined policies and compliance requirements. Furthermore, it continuously reviews the collected evidence to verify that policies, such as restricting access to an sFDP only to consumers from a designated organization and ensuring deployment at an authorized location, are upheld. It interacts with the Federation Operation Smart Contracts and Evidence Storage. The Federation Operations describe the smart contracts required for interacting in the federation, namely the Federation Smart Contract and the TNS Smart Contract. The Evidence Storage combines the functionality required for storing and retrieving evidence, including the IPFS as well as the DocumentStore Smart Contract. As shown in Figure 19, in the evidence-medicine pilot Advocate is installed on each of the hospital nodes that are exposing an sFDP.

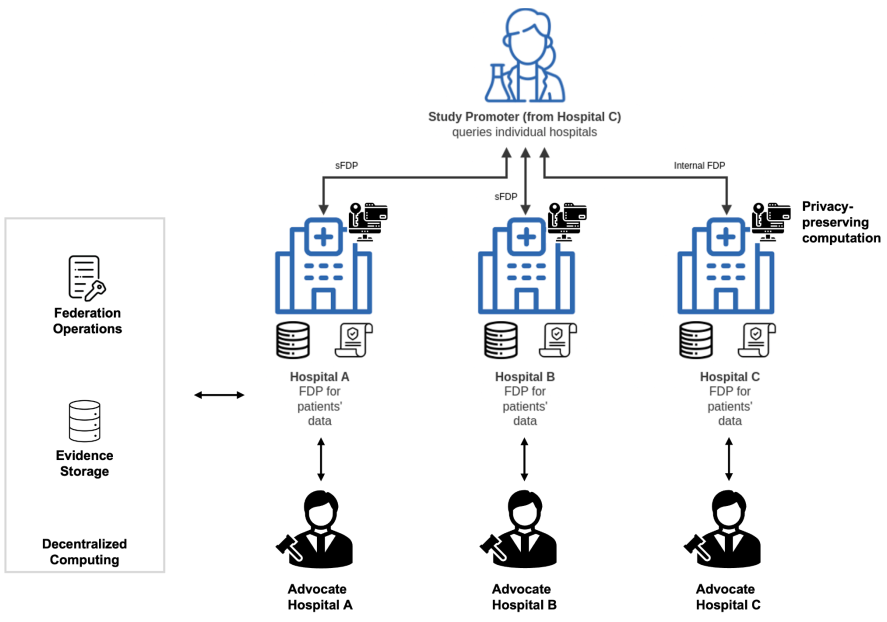


*Figure 20: : Trust plane in the evidence-based medicine pilot.*

**Sharemind MPC** technology is showcased, drawing inspiration from the Evidence-based Medicine pilot. In this scenario, three hospital nodes run an MPC protocol written in the SecreC language to collaboratively execute a query on private patient inputs while ensuring data confidentiality. Each hospital provides its own private statistical distributions regarding fasting blood glucose (FBG) levels from their internal datasets (which may have any size), and through secure, rule-based computations, the protocol generates a synthetic dataset consisting of N rows (e.g. 10 million). The scenario simulates a researcher requesting a dataset of FBG (mg/dL) observations for patients aged between 18 and 25 over a specified date range. The dataset contains patients ids, ages, observation dates, concept ids, observation value and unit id. This setup not only preserves the privacy of the patient data by never exposing raw inputs but also enables the researcher to work with a synthesized dataset that reflects the underlying distributions from each hospital. This demonstration shows how Sharemind MPC can enable privacy-preserving data analysis in healthcare, facilitating collaborative research without compromising patient confidentiality. Additionally, with this demonstration, we show the validation of KPI 2.3 that states the “Ability to set up privacy-preserving/confidential analytics from at least 3 members of a federation, with at least 10 million rows in the combined dataset.”

## ii.USE CASE PILOT #3: SMART VITICULTURE

The smart viticulture pilot showcases a data sharing scenario for collecting soil moisture information of multiple vineyards. Virtual edge devices, located at different vineyards, collect and store the data as part of a centralized data lake. The aggregated soil moisture data is exposed. A description of the smart viticulture pilot can be found in deliverable D2.1, D2.2 and D2.3 [7, 8, 9] under the “USE CASE PILOT #3: SMART VITICULTURE” chapter.

As shown in Figure 20, **Advocate** is installed on the node aggregating soil moisture data from the virtual edge devices. It allows to collect evidence for interactions related to sharing aggregation results to external customers. By closely observing the entire data-sharing lifecycle, Advocate ensures that all interactions adhere to predefined policies and compliance requirements. Furthermore, it continuously reviews the collected evidence to verify that policies, such as restricting access to the sFDP only to the vineyard owner are upheld. As already described in the first use case dependent pilot, the evidence-based medicine pilot,

Funded by the European Union

the TEADAL Trust Plane again relies on interaction with the Federation Operations and the Evidence Storage.

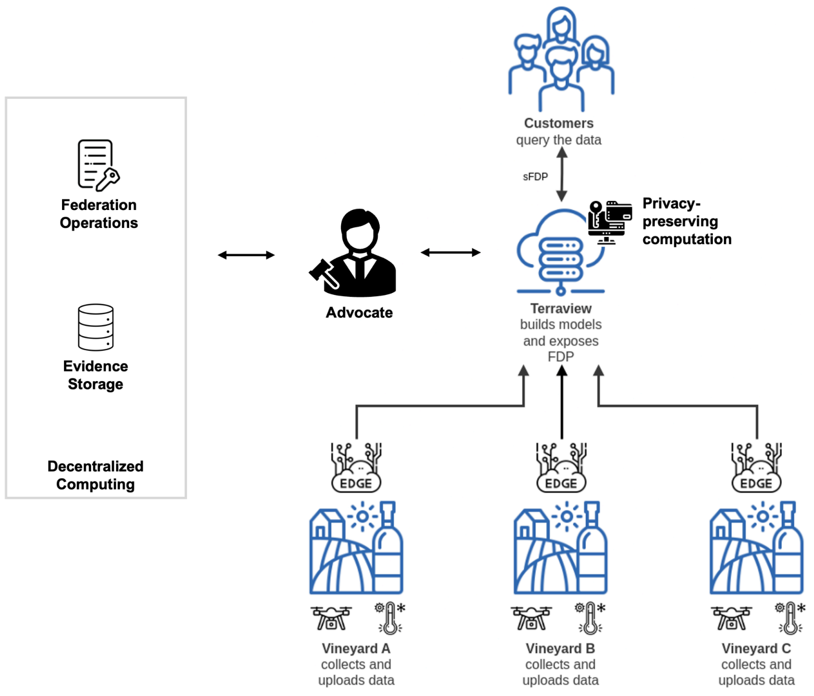


*FIGURE 21: TRUST PLANE IN THE VITICULTURE PILOT.*

## iii. USE CASE PILOT #4: INDUSTRY 4.0

The industry 4.0 pilot involves two ERT Group plants located in Portugal and the Czech Republic. Each of the plants calculates a set of Key Performance Indicators (KPIs), which are shared to calculate company-wide KPIs. The Czech plant functions only as a plant, while the Portuguese plant functions both as a plant and as headquarters. As such, the Portuguese plant collects its own KPIs and those of the Czech plant, stores the unified, company-wide KPIs and exposes them as FDP. A description of the industry 4.0 pilot can be found in deliverable D2.1, D2.2 and D2.3 [7, 8, 9] under the "USE CASE PILOT #4: INDUSTRY 4.0" chapter.

As shown in Figure 21, **Advocate** is installed on the ERT nodes of both the Czech and the Portuguese plant. It allows to collect evidence for interactions related to sharing plant-specific KPIs between the nodes and company-wide KPIs. This establishes a chain of trust between ERT plant operators and company management. By closely observing the entire data-sharing lifecycle, Advocate ensures that all interactions adhere to predefined policies and compliance requirements. Furthermore, it continuously reviews the collected evidence to verify that policies, such as restricting access to the sFDP only to the company management are upheld. As already described in the first use case specific pilot, the evidence-based medicine pilot, the TEADAL Trust Plane again relies on interaction with the Federation Operations and the Evidence Storage.

Funded by the European Union

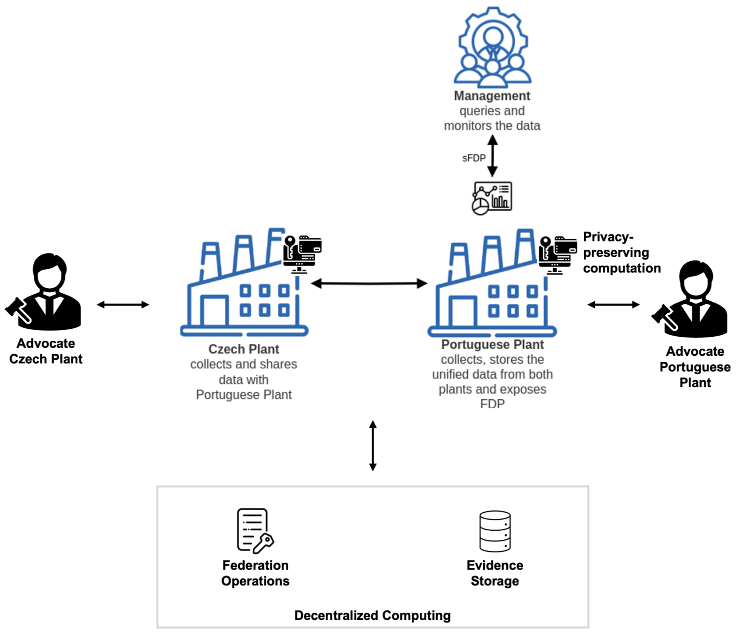

Management
queries and monitors the data
sFDP
Privacy-preserving computation
Advocate Czech Plant
Czech Plant
collects and shares data with Portuguese Plant
Portuguese Plant
collects, stores the unified data from both plants and exposes FDP
Advocate Portuguese Plant
Federation Operations
Evidence Storage
Decentralized Computing

*FIGURE 22: TRUST PLANE IN THE INDUSTRY 4.0 PILOT.*

## iv. NEW USE CASE PILOT: OPTIMISING FINANCIAL RETURNS FROM RENEWABLE ENERGY SOURCES

A financial focused integration was first described in the shared financial data governance pilot that can be found in deliverable D2.1 and D2.2 [8, 9] under the “USE CASE PILOT #5: SHARED FINANCIAL DATA GOVERNANCE” chapter. BOX2M replaces this financial data governance pilot, as described in D2.3 [7] under the chapter “NEW USE CASE PILOT: OPTIMISING FINANCIAL RETURNS FROM RENEWABLE ENERGY SOURCES”, to which we refer to in this deliverable.

**TEE-based pipelines** were implemented in the financial-focused pilot to securely process sensitive financial data while ensuring strict regulatory compliance. In these pipelines, confidential financial data is processed within VM-based Trusted Execution Environments—using Confidential Containers orchestrated via Kubernetes—which isolate and safeguard critical computations. Remote attestation generates cryptographic evidence of the secure execution environment. As detailed in D5.2 [2], this architecture not only protects sensitive financial operations from unauthorized access but also enhances transparency and auditability, ultimately providing stronger trust guarantees for distributed operations.

Funded by the European Union

# 4 Conclusions

This deliverable completes the third phase of establishing the TEADAL trust architecture. In this report, we focused on validating the trust mechanism provided by TEADAL and demonstrating its capabilities in use case independent and TEADAL specific use case dependent scenarios. Therefore, we revised the requirements gathered from the pilot descriptions and defined the three trust layers of identity, computational, and behavioral trust. Building on the previous results, we reviewed the trustworthy architecture with its components, focusing on the trust layers. We also explained how the various components and mechanisms apply to a federated environment.

Our evaluation demonstrates the capabilities of integrating tools and components such as Advocate or TrustOps, and having an end-to-end verifiable FDP, and covers several aspects of a validation, such as the cost of gas and experiments on the claim performance of Advocate.

Funded by the European Union

## REFERENCES

[1] TEADAL Consortium, “D5.1 TRUSTWORTHY DATA LAKES FEDERATION FIRST RELEASE REPORT,” 2023.

[2] TEADAL Consortium, “D5.2 TRUSTWORTHY DATA LAKES FEDERATION SECOND RELEASE REPORT,” 2024.

[3] S. Azzabi, Z. Alfughi, A. Oud,. “Data Lakes: A Survey of Concepts and Architectures,” in *Computers*, 2024.

[4] P. Germani, M. A. Manzari, R. Magni, P. Dibitonto, F. Previtali and E. D’Agostini, "Building Trustworthy AI Systems: AI Inference Verification with Blockchain and Zero-Knowledge Proofs," in *6th Conference on Blockchain Research & Applications for Innovative Networks and Services*, 2024.

[5] E. Brito, F. Castillo, P. Pullonen-Raudvere and S. Werner, “TrustOps: Continuously Building Trustworthy Software,” in *28th International Conference on Enterprise Design, Operations and Computing*, 2024.

[6] V. Nikolaenko, S. Ragsdale, J. Bonneau and D. Boneh, “Powers-of-Tau to the People: Decentralizing Setup Ceremonies,” in: *Applied Cryptography and Network Security*, 2024.

[7] TEADAL Consortium, “D2.3 PILOT CASES’ FINAL DESCRIPTION AND INTERMEDIATE ARCHITECTURE OF THE PLATFORM,” 2024.

[8] TEADAL Consortium, “D2.2 PILOT CASES’ INTERMEDIATE DESCRIPTION AND INITIAL ARCHITECTURE OF THE PLATFORM,” 2024.

[9] TEADAL Consortium, “D2.1 REQUIREMENTS OF THE PILOT CASES,” 2023.

[10] TEADAL Consortium, “D6.1 TESTBED DESIGN,” 2023.

[11]TEADAL Consortium, “D6.2 INTEGRATION REPORT,” 2024.

[12] TEADAL Consortium, “D4.2 STRETCHED DATA LAKES SECOND RELEASE REPORT,” 2024.

Funded by the European Union