D5.2
REWIRE Continuous Authorisation, Trust Management, Monitoring Mechanisms and Trust Evidence Collection

Project number:	101070627
Project acronym:	**REWIRE**
Project title:	REWiring the ComposItional Security VeRification and AssurancE of Systems of Systems Lifecycle
Project Start Date:	1st October, 2022
Duration:	36 months
Programme:	HORIZON-CL3-2021-CS-01
Deliverable Type:	Report
Reference Number:	HORIZON-CL3-2021-CS-01-101070627/ D5.2 / v1.0
Workpackage:	WP5
Actual Submission Date:	30th September, 2025
Responsible Organisation:	SURREY
Editor:	Yalan Wang
Dissemination Level:	Public
Revision:	1.0
Abstract:	Deliverable 5.2 presents the final version of the REWIRE Blockchain architecture and the technical means to achieve continuous authorisation and on/off chain data management. It provides a comprehensive analysis of the final design, reasons about the technical choices of the Secure Oracle composition (leveraging both Town Crier and FPC technologies) and then it highlights all the building blocks associated with detailed evaluation results.
Keywords:	Blockchain, Secure Oracles, Town Crier, Hyperledger Besu, Fabric Private Chaincode, AI-based Misbehaviour detection

The The project REWIRE has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101070627.

Editor

Yalan Wang (SURREY)

Contributors (ordered according to beneficiary numbers)

Sofianna Menesidou, Thanassis Giannetsos, Panagiotis Banavos (UBITECH)
Kaitai Liang, Zeshun Shi (TUD)
Spiros Kousouris, Sotiris Kousouris, Erifili Ichtiaroglou (S5)
Ilias Aliferis (UNIS)
Manolis Sourligas (LSF)
Jesus Sanchez
Yalan Wang, Liqun Chen (SURREY)

Disclaimer

Executive Summary

This deliverable is the second deliverable of the WP5 deliverable series and focuses on the **REWIRE Blockchain core artifact** and the selected technical means to achieve continuous authorization and on/off chain data management. The deliverable identifies the various data **workflows** and **data types** of the REWIRE Blockchain Infrastructure. Six distinct data flows have been identified namely the a) **Storing and Indexing Application-related Data**, b) **Storing and Indexing Trust-related Evidence**, c) **Application-related Data Query for Generating Threat Intelligence**, d) **Attestation-related Data Query for Generating Threat Intelligence**, e) **Security Policy Distribution & Enforcement** and f) **MUD Profile Flow**. Additionally, it documents all Blockchain-integrated entities such as the Threat Intelligence components, SW/FW Distribution Service, and the MUD Profile Server.

The deliverable also provides an in-depth analysis of the **Secure Oracle Layer** in REWIRE, which is built on three core technologies: **TownCrier**, **Hyperledger Besu**, and **Fabric Private Chaincode**. These tools collectively enable secure and private data interactions. However, limitations such as restricted interoperability, rigid configurations, and fragmented execution have been identified. To overcome these challenges, REWIRE is exploring integrated solutions like the Phala Network to enhance overall security, privacy, and flexibility. Furthermore, D5.2 describes the types and functions of smart contracts deployed across Besu, FPC, and TownCrier. These contracts are central to REWIRE's automation, governing secure data management, device operations, and policy enforcement within IoT ecosystems. D5.2 also documents the **specification details** of TownCrier, FPC and Security Context Broker, including also the implemented **Smart Contract functions**.

Moreover, the focus on this final release of the REWIRE Blockchain is the **benchmarking of the end-to-end data flows**. An initial benchmarking of the REWIRE Blockchain Architecture had been provided in D5.2 and D6.1 focused on evaluating the off-chain data storage and internal operations of read/write from/to the different ledgers. This deliverable focuses on the end-to-end evaluation of the application-related data flow, the attestation-related data flow, the SW Update flow and the MUD Profile flow, including the micro-benchmarking of the corresponding internal phases. Several **stress tests and scalability tests have been performed** proving the efficiency and scalability of the REWIRE Blockchain software stack. Overall, the latency observed during write operations in the REWIRE blockchain infrastructure is primarily due to the underlying PoA consensus mechanism. However, query operations demonstrate low latency and SCB proved an scalable orchestration component.

Finally, the deliverable concludes with the evaluation of the **AI-based Misbehaviour Detection Engine** (AIMDE), a threat intelligence component that can act as a supplementary trust source within a broader trust assessment framework, contributing to a more comprehensive understanding of IoT device trustworthiness. REWIRE's AIMDE demonstrates strong capabilities in analyzing agroclimatic sensor data to identify a wide range of anomalies from point anomalies (such as sudden soil humidity changes) to contextual anomalies (like unexpected variations in indoor climate). It employs advanced ML/DL models in both supervised and unsupervised settings. Experimental results from the Smart Cities use case confirm that AIMDE achieves high accuracy, excellent precision, and strong recall across various detection scenarios. Although there is potential to further enhance its sensitivity, AIMDE has proven to be a robust and reliable component for augmenting trust evaluation mechanisms.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Translation
ABAC	Attribute-Based Access Control
ABE	Attribute-Based Encryption
ACC	Assisted Cruise Control
AIC	Attestation Integrity Verification
AIMDE	AI-based Misbehavior Detection Engine
CC	Confidential Computing
CNN	Convolutional Neural Network
DL	Deep Learning
ECDSA	Elliptic Curve Digital Signature Algorithm
ELK	Elasticsearch, Logstash, Kibana
FPC	Fabric Private Chaincode
IF	Isolation Forest
LOF	Local Outlier Factor
ML	Machine Learning
MDS	Misbehaviour Detection System
MUD	Manufacturer Usage Description
PoA	Proof of Authority
SCB	Security Context Broker
SSI	Self-Sovereign Identity
SVC	Support Vector Classifier
TC	TownCrier
TCB	Trusted Computing Base
TEE	Trusted Execution Environment
UC	Use Cases
VC	Verifiable Credential
VP	Verifiable Presentation
WASM	WebAssembly
ZKP	Zero-Knowledge Proofs

Versioning and contribution history

Version	Date	Summary of changes	List of Contributors
v0.1	13.01.2025	Table of Contents & Allocation of tasks to the partners	Kaitai Liang (TUD), Sofianna Menesidou, Panagiotis Banavos (UBI)
v0.2	26.03.2025	Summary of the final version of the REWIRE Blockchain Architecture (Chapter 2)	Sofianna Menesidou, Panagiotis Banavos (UBI)
v0.3	11.04.2025	Final version of the REWIRE Secure Oracle Lyaer and technologies leveraged. Also, this includes a high-level analysis of more modular secure oracles considering future migration to such functionalities (Chapter 3)	Zeshun Shi, Kaitai Liang (TUD), Sofianna Menesidou, Panagiotis Banavos (UBI)
v0.4	29.04.2025	Final version of REWIRE smart contracts definition (Chapter 4)	Kaitai Liang, Zeshun Shi (TUD), Aliferis Ilias (UNIS), Sofianna Menesidou, Panagiotis Banavos (UBI) (UBI)
v0.5	08.05.2025	AI-based misbehaviour detection documentation and trust aware authorisation and authentication (Chpter 7)	Spiros Kousouris (S5), Aliferis Ilias (UNIS), Manolis Sourligas (LSF) Jesus Sanchez (ODINS), Yalan Wang, Liqun Chen (SURREY)
v0.6	13.05.2025	First version of the overall REWIRE Blockchain infrastructure evaluation (Chapter 6)	Kaitai Liang, Zeshun Shi (TUD), Aliferis Ilias (UNIS), Sofianna Menesidou, Panagiotis Banavos (UBI) (UBI)
v0.7	27.05.2025	Documentation of REWIRE Security Context Broker functional specifications (Chapter 5)	Sofianna Menesidou, Panagiotis Banavos (UBI), Zeshun Shi (TUD)
v0.72	12.06.2025	Final version of evaluation results considering the end-to-end measurement extraction in the context of the Smart Cities use case (Chapter 6)	Sofianna Menesidou, Panagiotis Banavos (UBI), Jesus Sanchez (ODINS), Zeshun Shi (TUD)
v0.75	20.06.2025	Updates provided by all partners in all chapters based on comments circulated by the technical management	All partners
v0.8	15.07.2025	Liqun Chen (SURREY), Annika Wilde (RUB)	Internal review
v0.9	21.07.2025	Refinements based on comments from internal review	Dimitris Karras, Thanasis Giannetsos (UBITECH)
v0.95	29.09.2025	All results documented throughout the deliverable were ratified also against the end-to-end experiments performed in the context of the envisioned use cases (D6.2). This was the reason behind the consortium opting to submit the final version of all deliverables on M36 where all results were available	Dimitris Karras, Thanasis Giannetsos (UBITECH)
v1.0	30/09/2025	Submission of deliverable	Thanassis Charemis (UBITECH)

Chapter 1

Introduction

1.1 Scope and Purpose

The scope of D5.2 is to delve into the **final version of the Blockchain Architecture of REWIRE** focused on the REWIRE **workflows** and **blockchain-based artifacts** that complement the REWIRE framework. In a nutshell, the REWIRE's Blockchain Architecture is a modular system designed to provide secure, privacy-preserving, and trustworthy data lifecycle management.

As already documented in previous deliverables, the REWIRE Blockchain Infrastructure operates during the runtime phase of the REWIRE framework, delivering a broad set of functionalities to securely manage the lifecycle of IoT deployments, tailored to various data types and workflow requirements. Therefore, this deliverable aims to document the core building blocks of the **Blockchain Architecture** and the corresponding **technology stack** (e.g., the Besu, TownCrier, Fabric Private Chaincode) that are part of the **REWIRE Secure Oracle Layer**. On top of that, an alternative highly promising solution, namely Phala, is discussed and considered for future research and adoption.

In addition, the deliverable documents the implemented **Smart Contracts**, which are necessary to manage and secure the entire REWIRE ecosystem. These contracts are deployed across the multiple REWIRE environments, including Besu, FPC, and the internally managed TownCrier enclave. Collectively, they ensure trust, access control, and verifiability throughout the system. The contracts are organized according to their functional roles and deployment environments, ensuring that essential operations, such as data validation, secure storage, access control, and update management, are cryptographically enforced and inherently verifiable. The deliverable also provides the implementation details of contracts and the exposed APIs. The core component that manages interactions with the Blockchain Infrastructure and serves as an interface to other internal REWIRE components is the **Security Context Broker**.

In this final release of the REWIRE Blockchain, the primary emphasis is on **benchmarking the end-to-end data flows**. While initial benchmarking of the REWIRE Blockchain Architecture, detailed in Deliverables D5.2 and D6.1, focused on assessing off-chain data storage and the internal read/write operations across various ledgers, this deliverable extends the evaluation to a broader scope. It concentrates on the complete evaluation of key data flows depicted as four scenarios, including application-related, attestation-related, SW update, and MUD profile flows, along with detailed micro-benchmarking of their internal phases. A series of stress and scalability tests have been conducted, demonstrating the efficiency and scalability of the REWIRE Blockchain software stack.

Last but not least, the deliverable provides the **evaluation** results of the **AI-based Misbehaviour Detection Engine** as a Threat Intelligence module designed to serve as an additional trust source within a comprehensive trust assessment framework, enhancing the evaluation of IoT device trustworthiness. AIMDE showcases advanced capabilities in processing agroclimatic sensor data to detect a diverse range of anomalies from point to contextual anomalies. It leverages sophisticated Machine Learning/Deep Learning (ML/DL) techniques in both supervised and unsupervised modes. Evaluation results from the Smart

Cities use case demonstrate that AIMDE consistently delivers high accuracy, strong precision, and reliable recall across multiple detection scenarios. While there remains room to improve its sensitivity in certain contexts, AIMDE has proven to be a dependable and effective component for reinforcing trust assessment in IoT environments.

1.2 Relation to other WPs and Deliverables

D5.2 delivers the 2nd and final version of the REWIRE Blockchain Architecture, continuous authorisation, and on/off chain data management. The 1st version were documented in D5.1 [5]. D5.2, apart from considering D5.1, takes as input the project's final architecture and the security, privacy and trust requirements that have been defined in D2.1 [4] and finalised in D2.2 [7]. In addition, as the Blockchain Infrastructure plays an important role in the SW/FW Distribution for the secure lifecycle management, auditing and certification of IoT ecosystems, it takes as input the SW/FW Update and Migration processes specifications and designs of WP3. The developments of WP4 regarding the definition of the capabilities of the REWIRE edge devices and the consideration of Keystone, are also given as input in D5.2. Finally, the output of WP5, will be given as input to WP6 for the delivery and evaluation of the final version of the REWIRE Integrated Framework. As can be seen in Figure 1.1, D5.2 will be used as the baseline of the Blockchain-based artefacts.

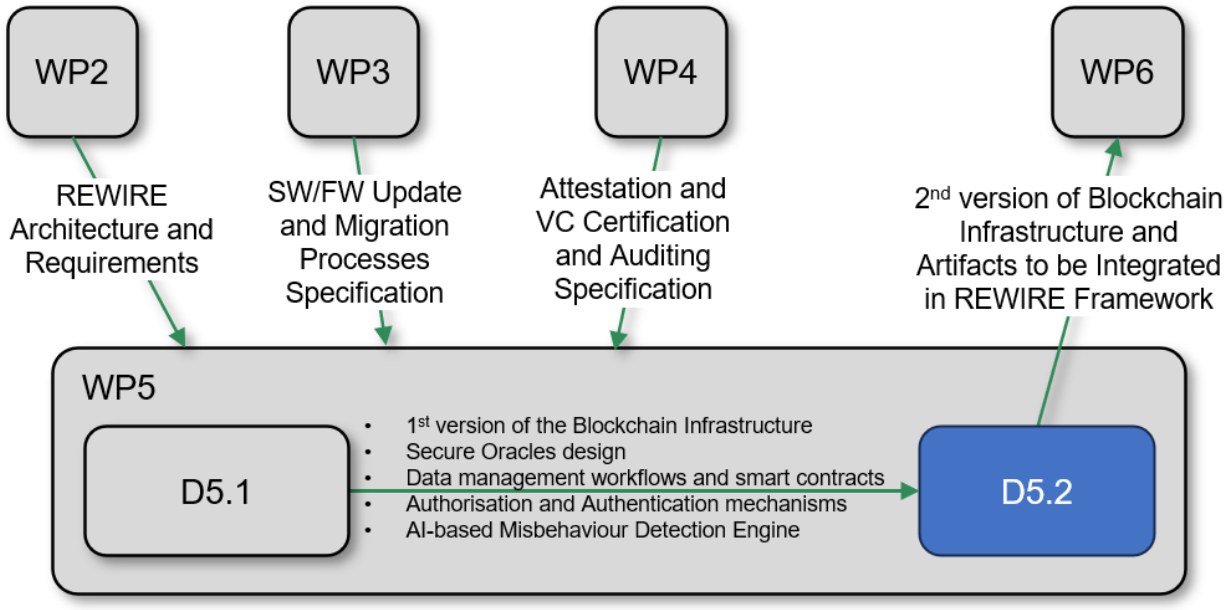

Figure 1.1: Relation of D5.2 with other WPs and Deliverables

1.3 Deliverable Structure

This deliverable is structured as follows.

Chapter 2 documents the final REWIRE Blockchain Architecture, including the architectural building blocks and the core data sharing workflows.

Chapter 3 highlights the selected technology stack of the REWIRE Secure Oracle Layer and discusses potential alternative solutions such as the Phala Network that could replace existing stack.

Chapter 4 focuses on the REWIRE Smart Contracts providing the different Smart Contract types, the implementation details including implemented functions, function types, parameters and interactions.

Chapter 5 provides an overview of the REWIRE's secure coordination layer, the Security Context Broker, its functional specification and implementation details.

Chapter 6 provides the evaluation results of the REWIRE Blockchain Infrastructure focusing on the core REWIRE workflows.

Chapter 7 focuses on the AI-based Misbehavior Detection Engine providing implementation details and evaluation results in the context of Smart Cities use case, proving its effectiveness as an additional trust source in the context of a trustworthiness framework.

Chapter 8 concludes the deliverable.

Annex provides examples of the a) MUD Profile, b) Attestation report and c) MSPL SW/FW Update policy.

Chapter 2

REWIRE Blockchain Architecture - Final Version

2.1 Architectural Building Blocks & Position in the Overall Compute Continuum

This section provides a description of the internal building blocks of **REWIRE Blockchain Infrastructure**, which is a **modular and layered system** designed to ensure secure, privacy-preserving, and trustworthy data lifecycle management. It integrates multiple trusted technologies and components, each fulfilling a distinct role in the data pipeline, from data ingestion and verification to storage and policy-enforced access.

At the core of this infrastructure lies the **Security Context Broker (SCB)**, which serves as the main orchestrator and entry point for all blockchain-related interactions. SCB manages query access, enforces fine-grained access control policies (e.g., ABAC), and initiates synchronous communication between subcomponents. Specifically, SCB emits blockchain events to trigger trusted workflows, including data fetching and attestation processing, acting as the coordinating hub between Besu, TownCrier, and Fabric Private Chaincode (FPC).

The **TownCrier Secure Oracle**, deployed in its standard configuration, acts as a bridge between blockchain logic and off-chain data sources. It continuously monitors Besu for specific on-chain events triggered by SCB. Once an event is detected, its **Relay component** forwards the request to the **SGX enclave**, where sensitive operations (such as HTTPS data fetching, verification, and signing) are performed securely. This ensures that data passed to the blockchain is both authentic and tamper-proof.

The **Besu Blockchain** functions as the event log and immutable ledger for external application and attestation data. It also serves as a coordination layer between SCB and TownCrier.

For storing sensitive attestation-related data, the system integrates **Fabric Private Chaincode (FPC)**. FPC supports **confidential smart contract execution** inside SGX enclaves, ensuring the **private and secure storage** of device attestation reports. It is used particularly in workflows where high assurance and privacy are required for data verification and long-term storage.

To handle large-scale data and overcome blockchain storage limitations, the infrastructure also includes a dedicated **Off-Chain Data Storage** component powered by the **ELK Stack (Elasticsearch, Logstash, Kibana)**. This hybrid architecture ensures that only integrity-protected references (e.g., hash pointers) are stored on-chain for datasets exceeding 20KB, while the actual data are indexed, stored, and queried off-chain via secure interfaces.

Finally, the system's external actors—such as **AI-based Misbehavior Detection engines, devices, and trust assessment services** interact with this infrastructure through SCB, leveraging its harmonized and access-controlled interfaces.

In nutshell, this architecture establishes a **layered trust model**, where (a) trusted hardware (Intel SGX) anchors the **execution integrity**, (b) secure enclaves (in TownCrier and FPC) ensure data **confidentiality** and **verifiability**, (c) the SCB manages orchestration and access control, (d) blockchain ledgers (Besu and FPC) provide **immutability** and **traceability** and (e) off-chain storage enables **scalability**. Together, these components form a cohesive and secure blockchain infrastructure that balances **performance**, **verifiability**, and **privacy** across distributed data workflows.

Last but not least, we also assume that the edge device is securely on-boarded as described in the Zero-Touch Onboarding Phase and thus has already acquired its VCs by the **Verifiable Credential Manager** enclave.

2.2 High-Level Execution of Data Sharing Agreements & Workflows

This section presents the final high-level REWIRE Blockchain reference architecture that highlights the workflows and updates from the previous version in D5.1 [5]. As already mentioned in previous REWIRE deliverables, the REWIRE Blockchain Infrastructure is part of the runtime phase of the REWIRE framework, where a wide range of functionalities are provided to support the secure life cycle of the IoT deployments focused on different data types and workflows. It conceptually supports two different ledgers, the Besu and the FPC, to accommodate the different levels of privacy requirements. Figure 2.1 presents the high-level REWIRE Blockchain architecture, where all the data-sharing agreements and workflows are depicted with different colours. The sections below describe these flows in detail.

2.2.1 Storing and Indexing Application-related Data

This workflow focuses on the **secure storing and indexing application-related data** —such as satellite telemetry— into the REWIRE Blockchain Infrastructure, leveraging the Besu Blockchain and the TownCrier Secure Oracle (**black arrows**). More specifically, the flow showcases how REWIRE leverages **event-based architecture** of Besu to **securely bridge off-chain data sources with the on-chain world** and the **trusted SGX enclave** of TownCrier, orchestrated under the control and business logic of the **Security Context Broker (SCB)**. The detailed steps of the flow are listed below:

(0) **Store application data** - External applications (e.g., satellites) running on edge-devices periodically push **application-related data** (e.g., telemetry, network and system data) and its associated **proof of knowledge** (e.g., VPs) into an InfluxDB database, either at defined time intervals or upon data readiness according to its local policy. As aforementioned, each stored data packet is bundled with its proof of knowledge (i.e., the VPs) that verify the origin and trustworthiness of each data source through cryptographically signed attributes (e.g., identity or device state attributes).

(1) **Emit application related data event** - The **SCB** periodically, either at defined time intervals or upon a specific triggering event, emits a data request event to the entire **Besu Blockchain** network.

(2) Then, the **Besu Blockchain**, receives this event and notifies the **TownCrier** oracle to fetch and validate the new application data stored on the InfluxDB. The **TownCrier** node, deployed in its standard architecture, is composed of a **Relay** component and a trusted **SGX enclave**, where the **Relay** continuously monitors **Besu** for such **SCB**-triggered events. **Relay** as a client to this **Besu Blockchain** network receives the signal, verifies that the signal instructs an operation dedicated from himself and initiates the fetching process from the InfluxDB.

(3) **Fetch application data & VPs** - Upon detecting a relevant event, the **TownCrier**'s **Relay** forwards the data request to the enclave to initiate a secure HTTPS channel with the InfluxDB for fetching the application data and the corresponding proof of knowledge. More specifically, the **TownCrier** queries InfluxDB to fetch the data using the implemented interfaces. REWIRE's implemented interfaces support two types of queries for fetching data either the new latest data or all the data after a specific time. (3a) **VPs** - The

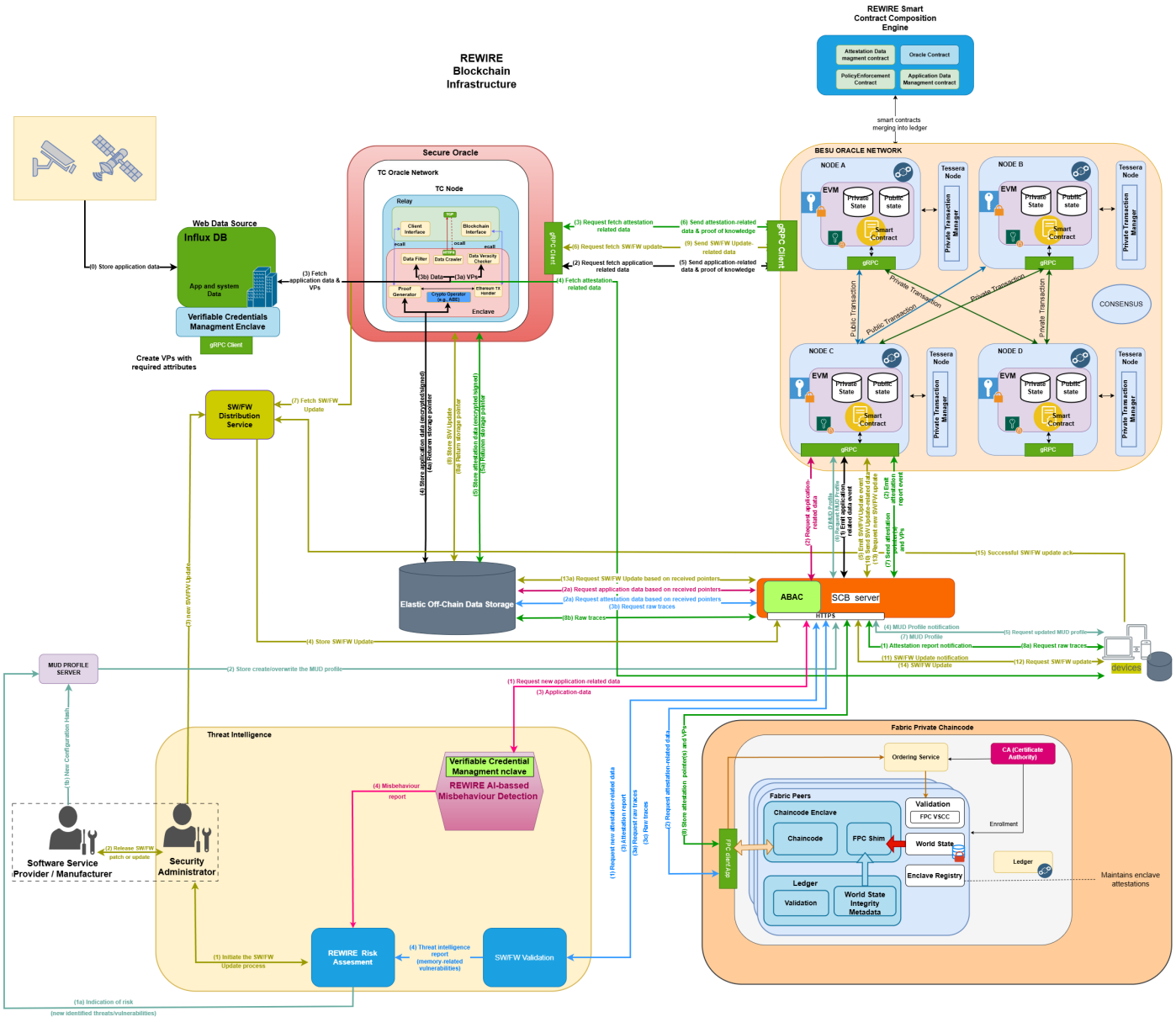

Figure 2.1: REWIRE Blockchain Infrastructure Architecture

enclave then verifies the proof of knowledge to ensure that the data was issued by an authenticated and trustworthy entity. This verification happens entirely within the SGX-protected environment, maintaining data confidentiality and preventing tampering. (3b) **Data** - After the successful verification and data veracity check, the **TownCrier** internally filters and normalises these application data in a structure readable by the AI-based Misbehaviour Detection Engine.

(4) **Store application data (encrypted/signed)** - Then the **TownCrier** stores the application-related data to the REWIRE Blockchain Infrastructure. If the dataset is **larger than 20KB**, it is securely stored in the **Elastic Off-chain Data Storage** (ELK Stack). Beforehand, the **Crypto ABE** sub-component is used to turn the application data in encrypted form (if such an operation is needed based on the security requirements), and the **Proof Generator** sub-component to store them with integrity and digitally signed (e.g., using the **TownCrier** key to perform signature). (4a) **Return storage pointer** - A cryptographically hashed pointer referencing the dataset is generated and returned to the **TownCrier**. The returned storage pointers are encrypted and signed by the **TownCrier** using the **Proof Generator** subcomponent.

(5) **Send application-related data & proof of knowledge** - Then the proof of knowledge along with either the **encrypted and signed off-chain pointers** (If the dataset is >**20KB**) or the actual data or the actual data (If the dataset is<=**20KB**), are pushed onto the **Besu** ledger as an immutable on-chain

record, ensuring auditable, verifiable, and tamper-resistant storage of application data.

2.2.2 Storing and Indexing Trust-related Evidence

This workflow focuses on the **secure storing and indexing trust-related data** (e.g., attestation reports) generated by edge devices into the REWIRE Blockchain Infrastructure, reinforcing trust and enabling scalable, policy-driven processing of verifiable device data (green arrows). Starting from **SCB** orchestration, through **TownCrier**'s enclave-based authenticity checks, to **FPC**'s attribute-based validation ensures that trust-related data are handled in a scalable, accountable and privacy-preserving manner. Thus, this flow is critical for maintaining trust across the REWIRE ecosystem. An example of attestation report is provided in the Annex 8.2.

REWIRE supports two modes of operation. A synchronous one, where the **SCB** has a policy and triggers the flow to store the latest attestation report periodically and an asynchronous one, where the **Facility Layer** of the device notifies the **SCB** indicating that the report is finalized and ready for consumption. The detailed steps of the flow are listed below:

(1) **Attestation report notification** - When an edge device completes one or more attestation reports, the **Facility Layer** of the device notifies the **SCB** indicating that the report is finalized and ready for consumption.

(2) **Emit attestation report event** - The **SCB**, acting as the orchestrator of secure data interactions in REWIRE, emits an attestation-specific event on the **Besu Blockchain** network, signalling the need to fetch the new attestation-related data. This event includes device-specific metadata and serves as a verifiable trigger to initiate the rest of the flow. As already mentioned, in the asynchronous mode of operation, this **SCB** emits this event triggered by the **Facility Layer** of the device, while in the synchronous mode **SCB** is triggered by a policy stored on the **SCB** periodically.

(3) **Request fetch attestation related data** - As with the previous flow, the **TownCrier** oracle, operating in its standard architecture, is composed of a Relay component that acts as Besu client and continuous monitors **Besu** for new events. Upon detecting the **SCB**-issued event stored to **Besu**, **TownCrier** activates its SGX enclave to securely fetch the attestation data directly from the target device (through the **Facility Layer**).

(4) **Fetch attestation related data** - The **Facility Layer** of the device responds with the attestation-related data and a proof of knowledge that serves as cryptographic proof of the report's origin and issuer. The enclave inside **TownCrier** performs the first stage of validation by verifying the authenticity of the proof of knowledge, ensuring that the data indeed comes from a valid, trusted device securely on-boarded during the ZTO phase. If this check fails, the attestation data is discarded. If successful, the attestation report is then evaluated for storage.

(5) **Store Attestation Data (encrypted/signed)** - After the successful verification and data veracity check, the **TownCrier** stores the attestate-related data to the REWIRE Blockchain Infrastructure. In this case, given the high frequency of incoming reports and the fact that they often exceed 20KB, attestation data is always stored in the **Elastic Off-chain Data Storage**. **TownCrier** securely stores the report in the ELK-based **Elastic Off-chain Data Storage** and generates a cryptographic hash pointer to reference the data. **TownCrier** may also encrypt the attestation-related data with **ABE** and sign them, as with the application-related data, prior storing off-chain. (5a) **Return storage pointer** - A cryptographically hashed pointer referencing the dataset is generated and returned to the **TownCrier**. The returned storage pointers are signed by the **TownCrier** using the **Proof Generator** subcomponent.

(6) **Send attestation-related data & proof of knowledge** - The storage pointer along with the verified proof of knowledge, is then immutably recorded on the **Besu** blockchain.

(7) **Send attestation pointer(s) and VPs** - Once the attestation-related data is successfully registered in **Besu**, the **SCB** —listening via WebSockets— is notified and initiates the process to store them in the **FPC**.

(8) **Store attestation pointer(s) and VPs** - After, **SCB** retrieves the pointer(s) and proof of knowledge, submits them to **FPC** for secure storage. Inside **FPC**'s own SGX enclave, a second verification process is performed. While **TownCrier** focuses on authenticity, the **FPC** validation step checks whether the proof of knowledge contains the correct set of attributes that classify the attestation as successful or failed, based on predefined policy constraints. If the proof of knowledge passes this check, the pointer and metadata are committed to **FPC**'s secure ledger. However, if the **FPC** validation fails, **SCB** initiates a remediation path: (8a) **Request raw traces** - it contacts the source device and requests the full raw trace data related to the failed attestation session. (8b) **Raw traces** - These raw traces are then also stored securely in the **Elastic Off-chain Data Storage** for future auditing or forensic investigation (e.g., by the Risk Assessment of the SW/FW Validation component).

2.2.3 Threat Intelligence Data Management

The threat intelligence data management includes two workflows. One **querying the application-related data** (pink arrows) by the **AI-based Misbehaviour Detection Engine** and another **querying the attestation-related data** (blue arrows) by the **SW/FW Validation Component**.

2.2.3.1 Application-related Data Query for Enabling Misbehavior Detection

This workflow focuses on the secure and policy-controlled **querying and consumption of application-related data** by external components for threat intelligence. In the context of REWIRE, the **AI-based Misbehavior Detection Engine**, is the primary consumer of the application-related data for generating behavior insights and anomaly reports. All external interactions with the REWIRE Blockchain Infrastructure are performed through the **SCB**, which serves as the central access point for external entities. These interactions are conducted via HTTPS to ensure confidentiality, integrity, and authentication.

It has to be noted that, prior to any data retrieval is permitted, the querying component (e.g., the AI-based Misbehavior Detection Engine) must first be on boarded to the REWIRE Infrastructure via the SCB (Zero-Touch Onboarding Phase). This phase includes the registration of the component on the Besu Blockchain as a recognized actor, accompanied by the creation and recording of a corresponding VC with the attributes describing the entity's permissions, role, and operational scope. The detailed steps of the flow are listed below:

(1) **Request new application-related data** - The **AI-based Misbehavior Detection Engine** initiates a data query by sending a request to the **SCB**, including a proof of knowledge of the attributes associated with the specific data set of interest. The **SCB** processes this query by first verifying the validity of the proof of knowledge encoded as a credential. If this is successful, then it is also ratified against the pre-defined **ABAC**) policy for accessing the dataset, ensuring that the requester possesses the necessary attributes to access the requested dataset (e.g., role = "misbehavior-detector", purpose = "anomaly-analysis").

(2) **Request application-related data** - If the component is authorized, the **SCB** proceeds to query the application-related data from the **Besu** Blockchain. (2a) **Request application data based on received pointers** - For big application data, this may involve also retrieving a pointer to off-chain storage and resolving the full data via the Elasticsearch SDK.

(3) **Application data** - The retrieved application data is returned securely to the querying component via HTTPS. At this point, the AI-based Misbehavior Detection Engine performs its computational logic—typically involving statistical models or ML-based inference—to analyze the satellite or other telemetry data for signs of misbehavior, deviation, or anomaly.

(4) **Misbehaviour report** - Then, the **AI-based Misbehavior Detection Engine** generates misbehavior detection reports, which are subsequently forwarded to the **REWIRE Risk Assessment** component for further evaluation. This allows for automated or semi-automated decision-making regarding risk levels, alert generation, or mitigation actions within the broader REWIRE system.

2.2.3.2 Attestation-related Data Query for Generating Threat Intelligence

This workflow focuses on the secure querying of attestation-related data by components tasked with analyzing SW and FW Vulnerabilities across edge devices. The primary actor in this scenario is the **SW/FW Validation** component, which relies on up-to-date attestation reports to identify misconfigurations, tampered firmware, or potential security risks in the device ecosystem. As with all external interactions in REWIRE, communications are performed through the **SCB**, acting as a centralized access orchestrator.

It has to be noted that, prior to any data retrieval is permitted, the querying component (e.g.,SW/FW Validation) must first be onboarded to the REWIRE Infrastructure via the **SCB** (Zero-Touch Onboarding Phase) and registered to the **FPC** ledger. This phase includes the registration of the component on the **FPC** Blockchain as a recognized actor, accompanied by the creation and recording of a corresponding VC with the attributes describing the entity's permissions, role, and operational scope. The detailed steps of the flow are listed below:

(1) **Request new attestation-related data** - To retrieve new attestation reports, the **SW/FW Validation** component submits a query request to the **SCB**, using HTTPS. As described in the previous flow, this request is accompanied by a proof of knowledge derived from its VC, which is then used in the **SCB**'s **ABAC** engine. This validation step checks whether the requester holds sufficient attributes—such as device classification, organizational role, or authorization context—to access the requested data.

(2) **Request attestation-related data** - If authorization checks are successful, the **SCB** triggers a query on the FPC ledger, retrieving the latest attestation report(s) associated with the relevant devices. These reports include information such as device states, firmware measurements, and verification outcomes, previously posted to FPC or storage pointers through the attestation storage flow. (2a) **Request attestation data based on received pointers** - For big attestation data, this may involve also retrieving a pointer to off-chain storage and resolving the full data via the Elasticsearch SDK.

(3) **Attestation report** - Once retrieved, the attestation report(s) are delivered securely to the **SW/FW Validation** component. Upon receiving the data, the component performs computational analysis to identify potential vulnerabilities, version mismatches, or configuration issues. (3a) **Request raw traces** - If the **SW/FW Validation** component identifies that the attestation report was not successful then it can proceed to request the raw traces by the **SCB**. (3b) **Request raw traces** - The **SCB** requests the raw traces stored to the **Elastic Off-chain Data Storage**. (3c) **Raw traces** - Finally, the **SCB** forwards back the raw traces to the **SW/FW Validation** component for further analysis.

(4) **Threat intelligence report (memory-related vulnerabilities)** - Based on this analysis, the component produces threat intelligence outputs such as identified memory-related vulnerabilities either existing CVEs or zero-day vulnerabilities, which are then forwarded to the **REWIRE Risk Assessment** component. These outputs contribute to the real-time risk posture evaluation of the system, enabling rapid remediation actions, alerting mechanisms, or compliance enforcement depending on the severity and context of the findings.

2.2.4 Security Policy Distribution & Enforcement

This workflow focuses on the secure distribution process of SW/FW updates to edge devices (ochre arrows), ensuring that critical updates are transmitted and applied in a verifiable and controlled manner, which is a core process of the secure life-cycle management of a device. An example of MSPL update policy is provided in the Annex 8.3. This flow leverages coordination among the **Security Administrator** who is aware of the potential threats generated by **REWIRE threat intelligence components** (e.g., **AI-based Misbehaviour Detection Engine** and **SW/FW Validation**), the **SW/FW Distribution Service**, the **SCB**, the **TownCrier** oracle, and the **Besu** Blockchain.

It is important to note that while **TownCrier** is the default mechanism for asserting data veracity and orchestrating secure update flows, it is also possible for the **SW/FW Distribution Service** to bypass

TownCrier and directly push updates to **Besu**. This path may be used in specific cases where tight integration with **TownCrier** is not required, or in scenarios where update integrity is managed through other mechanisms. This flexibility ensures that REWIRE can accommodate both autonomous enclave-driven workflows and trusted direct submissions when needed. The detailed steps of the flow are listed in sections below.

2.2.4.1 Release and Store SW/FW Update

(1) **Initiate the SW/FW Update process** - Following an indication of risk, the **REWIRE Risk Assessment** component originated by the REWIRE attestation enablers and the **SW/FW Validation** component or the **AI-based Misbehaviour Detection Engine**, the **Security Administrator** needs to deploy a patch or update. More specifically, the **Security Administrator**, after evaluating the REWIRE Risk Assessment report that aggregates all the threat intelligence information, initiates the SW/FW update process to enable secure over-the-air delivery of patches for IoT devices.

(2) **Release SW/FW patch or update** - The **Security Administrator** along with the **Software Service Provider** / **Manufacturer** create the patch or the update.

(3) **new SW/FW Update** - Then, the **Security Administrator** signs the update and sends it to the **SW/FW Distribution Service** responsible for managing devices, handling security patches/updates, and coordinating their deployment.

(4) **Store SW/FW Update** - The **SW/FW Distribution Service** encapsulates the update payload with relevant metadata, such as the targeted device(s), versioning, and deployment instructions and then signs and/or encrypts it. In the **one-to-many** scenario the **SW/FW Distribution Service** only signs the update with its private key. In the **one-to-one** scenario, signing and encryption is performed with the LRBC pre-shared key of the target device. Once the update is prepared, the **SW/FW Distribution Service** notifies the **SCB** to store this new SW/FW update.

(5) **Emit SW/FW Update event** - Acting as the orchestrator of secure data flows in the REWIRE system, the **SCB** emits a dedicated event on the **Besu** Blockchain to signal the need for secure data retrieval and logging of a new SW/FW update. This event serves as a trusted, immutable signal in the system's audit trail.

(6) **Request fetch SW/FW update** - The **TownCrier** oracle, which continuously monitors **Besu** via its **Relay** module, detects this event and securely contacts the **SW/FW Distribution Service** to fetch the SW/FW update along with its metadata.

(7) **Fetch SW/FW Update** - The **TownCrier**'s Relay forwards the data request to the enclave to initiate a secure HTTPS channel with the **SW/FW Distribution Service** for fetching the SW/FW patch or update along with its metadata.

(8) **Store SW Update** - Upon retrieval, the update is verified and processed inside **TownCrier**'s SGX enclave, ensuring its authenticity and integrity. The verified update is then stored to the **REWIRE Blockchain Infrastructure**. As with the previous flows, data greater than 20K which is the case for the SW/FW update are stored in the **Elastic Off-Chain Data Storage**. (8a) **Return storage pointer** - The corresponding storage pointers are returned back to be stored on the Besu Blockchain. At this stage, the SW/FW update is stored and indexed in the REWIRE Infrastructure.

(9) **Send SW/FW Update-related data** - The SW Update-related data (e.g., storage pointers of SW Update itself), is then immutably recorded on the **Besu** blockchain.

(10) **Send SW/FW Update-related data** - Following this, the **SCB** —subscribed to event updates via WebSockets— is notified of the successful update storage on-chain.

(11) **SW/FW Update notification** - The **SCB** retrieves the SW-update-related data and metadata from Besu and broadcasts the update to the appropriate edge device(s). Based on the mode of operation of the update, two different flows are defined (a) **the one-to-one** scenario and (b) the **one-to-many**.

2.2.4.2 Query and Enforce SW/FW Update

(12) **Request SW/FW update** - The **Facility Layer** of the notified device(s) interacts with the **Verifiable Credential Manager** to create the appropriate VP (through either the ABSC or the ZKP schemes described in D4.3) and then requests the specific update from the **SCB** providing the created VP for authorisation.

(13) **Request new SW/FW update** - The **SCB** checks the validity of the VP and then performs the ABAC. If successful, requests and receives the actual updated or the necessary pointers from the **Besu** Blockchain, which are used to retrieve the actual update for the specific device ID. As aforementioned, the SW/FW update in the **one-to-one** case is encrypted, while in the **one-to-many** is signed. (13a) **Request SW/FW Update based on received pointers** - The **SCB** requests and receives the SW/FW Update stored in the **Elastic Off-Chain Data Storage**.

(14) **SW/FW Update** - The **SCB** verifies and forwards the update to the device. The **Facility Layer** of the device(s) can then apply the update with cryptographic assurance regarding its origin and correctness. For instance in the (a) **one-to-one** scenario to decrypt and verify the update with the pre-shared LRBC key (this leverages the mirrored keys protocol described in D3.2 and D3.3, where it accesses the AE enclave gets the LRBC key) and in the (b) **one-to-many** scenario to verify the update, the ack will we be latter signed with the attestation key.

(15) **Successful SW/FW update ack** - After the update enforcement, the device performs a CIV and sends the acknowledgement, which is actually an attestation report, either encrypted (in one-to-one scenario) or signed (in one-to-many scenario) back to the **SW/FW Distribution Service**. The **SW/FW Distribution Service** will be able to check whether the update has been carried out successfully.

2.2.5 MUD Profile Flow

This workflow focuses on the MUD Profile lifecycle (a) form creation, storing and indexing into the REWIRE Blockchain Infrastructure, leveraging the **Besu** Blockchain, (b) to update and override its instance on the Blockchain and (c) to finally query of the profile by authorised devices (petrol arrows). An example of the MUD Profile is provided in the Annex 8.1.

2.2.5.1 Release and Store MUD Profile

(1a) **Indication of risk (new identified threats/vulnerabilities)** & (1b) **New Configuration Hash** - Following an indication of risk by the **REWIRE Risk Assessment** or a new configuration hash by the **Manufacturer**, the **MUD Profile Server** is notified to update the MUD profile for the specific device(s). More specifically, the **Manufacturer** is notified to release a new configuration with the new updated SW stack and to generate the new policy hash representing the new correct state of all target devices featuring the specific SW stack, which will use used as a new key restriction usage policy. Keep in mind that a field in the MUD Profile is the configuration hash that represents the expected configuration state for the key restriction usage during on-boarding. This field can also be updated dynamically during runtime (e.g., when an update is released).

(2) **Store create/overwrite the MUD profile** - Then the **MUD Profile Server** updates the MUD profile per type of device and notifies the **SCB** to store/update it in the REWIRE Blockchain Infrastructure. This includes the new threats/vulnerabilities or the updated configuration hash.

(3) **Store MUD Profile** - The **SCB** pushes the new MUD Profile to the **Besu** Blockchain.

(4) **MUD Profile notification** - The **SCB** notifies the **Facility Layer** of the edge device(s) that a new MUD Profile is released. Based on the characteristics included in the MUD Profile, the device(s) can determine whether the profile is intended for them.

2.2.5.2 Query MUD Profile

(5) **Request updated MUD Profile** - Upon notification, the edge device(s), check internally through its **Facility Layer** if it is intended for the specific device. If yes, the device initiates the key-restriction usage policy phase with the **Domain Manager**. More details are provided in D4.3 for binding the device's attestation key to the new key restriction usage policy represented by the new policy hash pushed into the updated MUD Profile. This step guarantees that only the Domain Manager as a trusted entity that can extract the policy hash form the uprated MUD Profile to enforce it to the target devices that are affected by this new policy hash. Finally, the **Facility Layer** of the device requests by the **SCB** the newly released MUD Profile.

(6) **Request new MUD Profile** - Then the **SCB**, after checking the VPs and performing ABAC successfully, requests for the specific edge **device** to get the MUD Profile stored on the **Besu** ledger.

(7) **MUD Profile** - Finally, the **SCB** forwards the corresponding MUD profile to the **device**.

2.3 Other Blockchain-dependent Entities and Artefacts

In addition to the core components of the REWIRE Blockchain infrastructure, namely the Security Context Broker (SCB), the Besu Blockchain, the TownCrier Secure Oracle, and the Fabric Private Chaincode (FPC), several supporting entities and artifacts rely on Blockchain functionalities to enable secure, traceable, and policy-driven operations. These entities are either directly interfacing with the Blockchain or indirectly leveraging it via SCB-mediated workflows. This section refers briefly to these REWIRE's Blockchain-dependent entities and artefacts of the REWIRE ecosystem in order to complete the description of the Blockchain Architecture of Figure 2.1.

⇒ **THREAT INTELLIGENCE**

- AI-BASED MISBEHAVIOR DETECTION ENGINE
  This Misbehaviour Detection Service (MDS) consumes verifiable application-related data (e.g., satellite metrics or other sensor measurements) fetched through the Secure Oracle Layer. Its interaction with the Blockchain is mediated through the SCB. Before querying data, the component undergoes Attribute-Based Access Control (ABAC) checks, using a Verifiable Presentation (VP). Once authenticated, it retrieves data—either directly from Besu or from the Off-Chain Data Storage (via a blockchain-logged pointer) and performs computations to detect anomalies or patterns indicating device misbehavior. The output is typically a risk flag or detection report, which is forwarded to the REWIRE Risk Assessment component.

- SW/FW VALIDATION
  Operating as a consumer of attestation data stored in the FPC, this component is also registered via SCB and is subject to ABAC checks enforced through the user's VP. Upon successful authorization, it retrieves one or more attestation reports associated with specific devices. The analysis performed includes signature validation, integrity verification, and vulnerability pattern recognition. The results are then used to inform remediation processes or trigger risk notifications. This flow ensures secure and privacy-preserving access to potentially sensitive device security data.

- REWIRE RISK ASSESSMENT
  Operating as a consumer of both AI-Based Misbehavior Detection Engine's and SW/FW Validation's component is the core component that initiates the SW/FW Update process in REWIRE and in a semi-supervised way with the help of the Security Administrator. The REWIRE Risk Assessment Engine does not have direct interaction with the Blockchain infrastructure but it receives the risk indicators generated by the aforementioned blockchain-depended components.

⇒ **SW/FW DISTRIBUTION SERVICE**
The SW/FW Distribution Servuce generate or acquire SW/FW Updates that are broadcast to edge devices. Updates can be introduced into the REWIRE Infrastructure either via SCB–TownCrier–Besu interaction or, in some cases, directly through SCB to Besu without the need for oracle validation. Metadata about the updates, such as target devices, versioning, and distribution scope, are immutably recorded on the blockchain. This provides auditability, tamper-resistance, and synchronization guarantees across the infrastructure.

⇒ **VERIFIABLE CREDENTIALS (VCS) AND VERIFIABLE PRESENTATIONS (VPS)**
These identity and access artifacts underpin the REWIRE trust model. VCs are issued to entities (devices or users) during onboarding and encode critical attributes. VPs are derived from VCs and attached to any interaction request—whether it's to fetch data, query a contract, or trigger a workflow. Their validation, in some cases, happens inside SGX enclaves (TownCrier) or within smart contract logic (FPC), and they are crucial for enforcing decentralized access policies aligned with W3C SSI standards.

⇒ **ELASTIC OFF-CHAIN DATA STORAGE**
Large data (e.g., attestation traces or application telemetry) is stored in the off-chain ELK stack. Blockchain-based pointers referencing these data are cryptographically hashed and stored either in Besu (for application data) or FPC (for attestation data). In future versions, these pointers are encrypted using Attribute-Based Encryption (ABE) to bind their accessibility to entities holding specific attributes in their VPs, ensuring an additional layer of confidentiality and fine-grained control.

⇒ **MUD PROFILE SERVER AS PART OF ZTO SCHEME AND MUD PROFILE CONSUMERS**
The REWIRE Blockchain Infrastructure enables advanced security policy management and secure lifecycle management of IoT devices. A core REWIRE offering is the Zero-Touch Onboarding (ZTO) mechanism and protocol, where the edge device that wants to join the REWIRE framework, must first authenticate to the MUD Profile Server to activate the cryptographic keys that will unlock the rest of the functionalities for the device. MUD Profiles describe intended network behaviors of devices and are stored and/or queried via SCB interfaces, with associated references or hashes optionally anchored on-chain for traceability. When other actors in the system (e.g., security monitors) need to retrieve a MUD profile, they issue ABAC-controlled queries through the SCB.

Chapter 3

REWIRE Secure Oracle Layer

3.1 REWIRE Technology Stack & Limitations

REWIRE's **Secure Oracle Layer** currently comprises two key technologies: (a) **TownCrier (TC)** and (b) **Fabric Private Chaincode (FPC)**. Both components rely on Intel SGX enclaves and collectively serve as REWIRE's **verifiable and confidential trust anchors** for enabling the trustworthy execution of (on-chain) data sharing agreements. This translates to having strong evidence on the **data veracity** of the various, interoperable data sources: By converging the use of REWIRE trust extensions throughout all layers of a system's/network's application stack (from the far-edge to the backend distributed data management infrastructure/ledger), REWIRE aims to ensure trust and trustworthiness of all actors in the continuum. Town Crier is used to securely fetch and verify off-chain data inside an SGX enclave, while FPC operates within a Hyperledger Fabric-based environment to perform **privacy-preserving logic execution and secure attestation data storage**. Although Hyperledger Besu and the Security Context Broker (SCB) are also critical components of the REWIRE Blockchain Infrastructure, they are not part of the Secure Oracle Layer per se. Instead, Besu acts as a synchronization and anchoring ledger, enabling inter-oracle coordination, while SCB provides policy enforcement, identity validation, and orchestration of workflows. The need for this intermediate communication layer stems from the current inability of TC and FPC to directly exchange verifiable data, since their underline trusted software is built in different and incompatible ways (divergent secure execution models).

While this architecture satisfies the initial privacy, verifiability, and data integrity requirements outlined in Deliverables D2.1 [4] and D2.2 [7] for the execution of secure and privacy-preserving data sharing agreements, it was intended as a transitional solution toward a **harmonized Secure Oracle framework**. By "harmonized" we envision a unified architecture that defines the necessary interfaces, tools, and communication protocols to enable direct integration and convergence of different secure oracle technologies, eliminating the need for an intermediary ledger such as Besu. As will be further elaborated in Chapter 6, the current architecture of REWIRE, in order to support this harmonization considering the fragmentation in the execution models necessitates the adoption of Besu for the interaction between the two secure oracle components (TC and FPC), introducing additional latency and architectural complexity. For example, TC is implemented in C++ using the legacy Intel SGX SDK with asynchronous event handling, while FPC follows the Hyperledger chaincode model, developed in Go and C++, and integrated with Fabric's endorsement flow. These differences limit interoperability and unified policy enforcement and increase development overhead when implementing cross-oracle communication logic. Further, during our implementation a number of technical constraints have been identified and are listed below:

- **Limited Interoperability**: The two enclaves (TC and FPC) operate under different execution environments and SDKs, which complicates orchestration and limits unified trust model.
- **Static User Configuration**: Dynamic user addition is not natively supported, particularly in FPC, which limits long-term scalability.

- **Static User Configuration and Limited Cryptographic Flexibility**:Dynamic user management, particularly in FPC, is not natively supported. Furthermore, advanced cryptographic primitives such as Attribute-Based Encryption, Zero-Knowledge Proofs and threshold signatures are not integrated into the current execution stack.
- **Fragmented Secure Execution Models**: The separation of trust responsibilities between TC and FPC leads to increased resource fragmentation and hinders the realization of a unified Trusted Computing Base.

In response to these limitations, REWIRE has initiated a dedicated research track into more unified alternatives, most notably the **Phala Network**[1], to explore whether a single decentralized platform can offer the security, confidentiality, privacy, and flexibility needed. Phala Network offers improved enclave abstraction, multi-oracle interoperability, and native support for confidential, privacy-preserving smart contract execution. This research path forms the foundation for future iterations of REWIRE's trust infrastructure and is presented as a candidate solution to mitigate many of the identified fragmentation and integration challenges.

3.2 Understanding Phala Network: Convergence of Confidential Computing with High Degree of Interoperability

The Phala Network represents a next-generation, **decentralized confidential computing platform**, designed to execute **privacy-preserving smart contracts** within TEEs. In the context of REWIRE's Secure Oracle Layer architecture, Phala could offer a highly promising solution to overcome several key limitations currently faced by the existing stack of TC and FPC. Unlike the current REWIRE stack, which delegates verifiable data fetching to TC, decentralized logging to Besu and sensitive data management to FPC, Phala **could unify these functionalities into a single framework**. This consolidation is made possible by Phala's architecture, where all **computation** is executed **off-chain within Intel SGX-enabled nodes** (called "Workers"), while interactions and trust proofs are synchronized with an on-chain Substrate-based consensus. Substrate-based consensus refers to the use of Substrate, a modular blockchain development framework, which powers the **Polkadot**[2] ecosystem. Substrate allows to build highly customized blockchains, runtime upgrades, and efficient state transition models. In Phala, Substrate is used to coordinate trust between Workers, anchor computation results, manage access control policies, and ensure that all confidential computation events are cryptographically verified and traceable. This combination of off-chain confidential execution and on-chain verifiability and coordination allows Phala to deliver a harmonized, interoperable, and scalable secure oracle framework. As a result, Phala addresses many of the technical bottlenecks identified in the REWIRE stack, such as:

- **Unified TEE execution models**, avoiding fragmentation across enclave SDKs or chaincode interfaces.
- **Native support for cryptographic agility**, including zero-knowledge proofs, threshold signatures, and attestation workflows.
- **Decentralized worker orchestration**, eliminating the need for intermediary components like Besu or SCB for coordination.
- **Secure, real-time oracle communication**, which aligns directly with the long-term vision for REWIRE's harmonized secure oracle layer.

[1] https://phala.network
[2] https://polkadot.com

3.2.1 System Architecture & Key Offerings

Phala's architecture is centered around confidential task execution, cryptographic policy enforcement, and secure interoperability. It introduces a decentralized trust model composed of three primary actors:

- **Clients**: These are external systems or users that submit encrypted data payloads or smart contracts (called Fat Contracts) for secure execution. → *In the context of REWIRE, clients could include components like the AI-based misbehavior detection module, the SCB, or even edge devices submitting telemetry or attestation data.*
- **Gatekeepers**: Specialized nodes responsible for session key management, attestation verification, and lifecycle operations such as key rotation or revocation. They also serve as the coordination interface with the Substrate-based blockchain, ensuring that all secure computation remains traceable and governed under consensus. → *While not directly present in REWIRE, Gatekeepers conceptually resemble elements of the Town Crier relay and SCB—components responsible for enforcing the trust interface between untrusted clients and trusted enclaves. However, unlike the REWIRE architecture, Gatekeepers in Phala are fully decentralized and consensus-aligned actors.*
- **Workers**: SGX-enabled compute nodes that execute Fat Contracts inside hardware-isolated enclaves. Workers are responsible for the actual secure execution of smart contracts, preserving the confidentiality of inputs and outputs—even from the host operating system. → *These Workers closely align with the REWIRE secure enclaves in Town Crier (for verified data acquisition) and FPC (for confidential storage and logic execution). In Phala, this functionality is unified under a common, programmable contract model.*

This architectural model satisfies REWIRE's requirements for **privacy-preserving processing**, **verifiable trust guarantees**, and **decentralized control** over sensitive data flows. In particular:

- **Workers** operate in a decoupled and parallelized fashion, making them suitable for high-volume attestation or telemetry processing.
- **Gatekeepers** enforce strict trust boundaries, ensuring that only remote-attested, policy-compliant Workers are allowed to participate—reflecting REWIRE's goals around dynamic trust characterization and runtime policy enforcement.
- Phala supports **compatibility with WebAssembly (WASM)** and, via bridging, **EVM-based environments**, enabling interoperability with legacy REWIRE components, such as Besu. → *This offers a migration pathway where REWIRE's current blockchain components (e.g., Besu) could remain in place temporarily while Phala is phased in as a unified oracle and compute layer.*

Importantly, Phala's confidential execution model ensures that sensitive computations—such as **identity verification, trust scoring**, and **dynamic policy decisions**, are performed with **end-to-end encryption** and **remote attestation**. This makes Phala not only a candidate technology, but a potential architectural evolution toward the **harmonized Secure Oracle Layer** envisioned in REWIRE.

3.2.2 Phala Blockchain: Confidential Smart Contract Execution

The Phala blockchain adopts a dual-layer architecture comprising a Substrate-based ledger and a decentralized Confidential Computing (CC) network. The ledger handles consensus and metadata anchoring, while the compute layer performs privacy-preserving contract execution inside Intel SGX-backed enclaves. Smart contracts in Phala, called Fat Contracts, are compiled into WebAssembly (WASM) binaries and executed entirely off-chain within secure enclaves, allowing sensitive logic and data to be processed without exposure to the host environment. In parallel, Phala integrates Ink! contracts, a domain-specific

smart contract language designed for Substrate. Ink! contracts are deployed on-chain and provide the interface for anchoring cryptographic commitments or state updates resulting from off-chain confidential computation. This separation enables Phala to decouple secure logic execution from ledger consensus while maintaining end-to-end verifiability. **It is important to clarify that in the context of REWIRE, Phala is not part of the current deployed architecture**. Rather, it is conceptually analyzed here as a potential alternative to overcome fragmentation and orchestration complexity in the existing secure oracle layer (composed of Town Crier, Besu, and FPC). The following execution flow outlines how core REWIRE operations, if mapped onto Phala, could be implemented within this unified, confidential execution framework:

1. **Encrypted Submission** - A REWIRE component (e.g., the SCB) submits an encrypted datagram to a deployed Fat Contract. The payload may include Verifiable Presentations (VPs), attestation logs, or policy decision inputs. This is analogous to SCB triggering the TC oracle in the current architecture.

2. **Routing via Gatekeepers** - The encrypted payload is routed through Phala Gatekeepers, which verify the Worker enclave's attestation, manage encryption session keys, and securely assign the task to an authorized SGX-backed Worker node. While this has no direct 1:1 equivalent in REWIRE, it conceptually overlaps with the functions of the SCB and TC relay.

3. **Secure Execution by Worker** - The Worker executes the Fat Contract inside a TEE. Tasks may include verifying credentials, evaluating ABAC policies, or performing cryptographic operations. This maps to the enclave-based logic currently handled separately in TC (for ingestion) and FPC (for secure attestation storage).

4. **Result Processing and Ledger Anchoring** - Depending on the classification of the result: (a) For small, non-sensitive results, an Ink! contract is used to store data directly on-chain and emit events. (b) For large or confidential outputs, a cryptographic hash or secure pointer is stored instead. This design mirrors how REWIRE uses Besu for event triggering and off-chain references while addressing current scalability constraints.

5. **Event Notification and Callback Integration** - The Ink! contract emits events once anchoring is complete. These can be consumed by REWIRE components such as the SCB or device managers to trigger downstream workflows (e.g., patch propagation or trust updates).

This end-to-end flow directly demonstrates that **Phala could serve as a functional equivalent** to REWIRE's current multi-oracle infrastructure. It provides a unified and privacy-preserving framework that could eliminate the need for separate oracle components (TC, FPC), bespoke relays, and custom synchronization mechanisms.

Moreover, by decoupling consensus from confidential computation, Phala offloads resource-intensive cryptographic operations, such as batch VP verification or ABAC enforcement—outside the block confirmation path. This has direct relevance for REWIRE's requirements around sub-second query latencies and mission-critical responsiveness, as shown in Chapter 6.

In summary, while Phala is not integrated into REWIRE's deployed infrastructure, its model and capabilities represent a strong candidate for future iterations. As discussed throughout this section, Phala could pave the way for a harmonized secure oracle layer, improving scalability, reducing integration overhead, and fulfilling REWIRE's vision of trust-verifiable, decentralized data processing in edge-critical environments.

3.2.3 Attestation, Sealing Keys, and Crypto Agility Layer

A core strength of the Phala architecture lies in its native support for remote attestation, sealing key policies, and crypto agility, which together offer the foundation for secure, privacy-preserving execution and decentralized identity enforcement.

Remote attestation: Remote attestation is a built-in requirement for every Worker node participating in Phala's compute network. Each Worker must present a valid enclave quote, demonstrating its integrity, origin, and configuration, before being authorized to handle confidential execution tasks. This principle is aligned with REWIRE's current use of attestation in TC and FPC, but Phala enhances the process by making attestation a network-level gatekeeping mechanism rather than a local validation step.

Sealing key policies: Sealing key policies is another core capability of TEEs, including REWIRE's current stack. However, in practice, access and management of sealing keys in existing components such as TC or FPC are static and tightly coupled to their respective enclave implementations. This makes fine-grained policy tuning or dynamic reconfiguration difficult, if not infeasible, without significant re-engineering. In contrast, Phala exposes sealing as a configurable primitive, allowing encrypted data to be tightly bound to the enclave's identity, contract scope, and contextual runtime. This facilitates secure continuity for tasks like long-term storage of attestation records, while also enabling state recovery under controlled, auditable conditions.

Crypto agility: Last but not least, Phala demonstrates competitive advantage through its crypto agility layer, embedded within its WebAssembly-based contract engine. Developers can directly implement or integrate a wide range of cryptographic operations within Fat Contracts, such as (a) ECDSA signature verification (aligned with W3C DID-SSI models), (b) Attribute-Based Encryption (ABE) for binding data to specific policy attributes, (c) threshold signatures or multi-party computation and (d) zero-knowledge proofs for selective disclosure.

While TEE-based components in REWIRE theoretically support such primitives, their integration and runtime configurability are extremely limited by SDK constraints and rigid enclave design. In contrast, Phala makes cryptographic extensibility part of its core execution model, enabling more sophisticated trust workflows with runtime-defined behavior. This gives REWIRE a key architectural advantage. For instance, encrypted off-chain data pointers could be bound to specific user attributes and decrypted only by parties whose VPs satisfy those policies. However, this level of attribute-based encryption and contract-governed cryptographic control is currently out of the scope of TC and FPC. In conclusion, Phala provides a fine-tunable, extensible, and secure execution environment that addresses key limitations identified in REWIRE's current stack. It not only supports the harmonization of the currently fragmented secure oracle components, but also the evolution of REWIRE's data governance model toward cryptographic adaptability and long-term confidentiality. More technical insights are further discussed in Deliverable [7].

3.3 REWIRE Data Management Computing Tasks

The vision of REWIRE's secure oracle layer has always been centered around trust, privacy, and verifiability of data interactions. The current implementation, featuring TC for verifiable off-chain data acquisition, Hyperledger Besu for permissioned on-chain anchoring, and FPC for confidential attestation workflows, fulfils these goals through a federated multi-oracle design. However, it introduces significant architectural complexity, along with fragmented trust boundaries and limited extensibility across enclaves and runtimes.

In this context, the exploration of a Phala-based blockchain architecture with embedded secure oracle capabilities has emerged as a potential research direction. Rather than removing the need for discrete components, the goal is to evaluate whether Phala's unified design can serve as an equivalent abstraction to a multi-oracle environment capable of supporting the same trust, privacy, and verifiability requirements,

but with reduced orchestration overhead. In a nutshell, Phala's confidential computing model offers native support for (a) fetching and verifying trusted application data, (b) executing identity-authenticated, policy-enforced attestation workflows and (c) performing computation over encrypted datasets in secure enclaves. These functionalities, if proven scalable, could provide a cohesive trust execution substrate, where secure data consumption, evaluation, and output generation are governed within a single substrate-based system. This would allow for the convergence of oracle responsibilities under a harmonized, enclave-backed trust environment, potentially simplifying the enforcement of REWIRE's strict security and access control policies.

That said, this remains an ongoing research. The true scalability and throughput characteristics of such an approach, especially under high transaction volumes or large-scale concurrent device interactions, must further evaluated. The adoption of a Phala-based model could therefore act as a next-phase evolution for REWIRE's Blockchain architecture, informing follow-up initiatives and research activities beyond the current work package (WP5). This direction is further contextualized in the REWIRE research roadmap, where the unification of trusted execution and verifiable storage continues to be a key architectural challenge.

Chapter 4

Smart Contract in REWIRE

The REWIRE project implements a sophisticated blockchain-based architecture where smart contracts play a pivotal role in managing and securing the entire ecosystem. These contracts serve as the backbone for automated operations, data management, and security enforcement within the network. Through the implementation of various specialized smart contracts, REWIRE establishes a robust framework for secure device management, data handling, and policy enforcement in IoT environments.

4.1 Types of Smart Contracts in REWIRE

In the REWIRE blockchain infrastructure, multiple types of smart contracts are implemented across different blockchain technologies to support distinct but interlinked functions. These contracts are deployed on **Besu (EVM-based blockchain)** and **FPC (Hyperledger Fabric-based, SGX-enabled private chaincode)** environments, as well as **internally managed within the TownCrier enclave**, and together they enforce trust, access control, and verifiability across the system.

These contracts are categorized based on their functional role and their deployment context, ensuring that critical operations like data validation, secure storage, access control, and update management are cryptographically enforced and verifiable by design. Table 4.1 below summarise the smart contract types in REWIRE.

⇒ **APPLICATION CONTRACTS (BESU)**
These contracts manage and log application-related data (e.g., satellite telemetry). SCB pushes application data through Besu-based contracts after the TownCrier verifies their integrity and origin. They also maintain cryptographic pointers to off-chain datasets when necessary.

⇒ **SW/FW UPDATE CONTRACTS (BESU)**
SW/FW update contracts are responsible for broadcasting firmware/software updates to edge devices. These contracts log update metadata, version information, and associated digital proofs, and coordinate with SCB and TownCrier to fetch updates and distribute them securely.

⇒ **POLICY CONTRACTS (BESU)**
Policy contracts enforce attribute-based access control rules in a transparent and tamper-evident way. They specify the access rights based on identities and Verifiable Credentials (VCs) and are referenced during access checks handled by SCB.

⇒ **SIGNATURE VERIFICATION CONTRACTS (BESU & FPC)**
In both chains, signature verification is essential. On Besu, they verify the digital signatures accompanying data to confirm authenticity, while in FPC, signature verification happens within the SGX enclave for

attestation-related operations, supporting ECDSA-based signatures aligned with W3C-SSI standards.

⇒ **ATTESTATION CONTRACTS (FPC)**
These contracts securely manage the lifecycle of attestation reports, including secure storage, signature validation, and retrieval by authorized entities. SCB mediates access to these contracts, ensuring strict enforcement of policies.

⇒ **TOWNCRIER CONTRACTS (INTERNAL, EVM-TRIGGERED)**
Although not deployed directly on-chain, TownCrier contracts operate in coordination with Besu contracts. They are triggered by events emitted from Besu and operate within the TownCrier SGX enclave to (a) fetch data from off-chain trusted sources, (b) verify Verifiable Presentations (VPs) and (c) push verified data back to Besu for storage or further processing.

⇒ **MUD PROFILE CONTRACTS (BESU)**
These contracts are responsible for managing MUD Profiles associated with specific device types. These contracts are deployed in Besu for (a) creation, (b) update and (c) search MUD Profiles.

Table 4.1: Smart Contract Types in REWIRE

Type	Technology	Description
Application Contracts	Besu	Log application data and pointers to off-chain sources after TC validation
SW/FW Update Contracts	Besu	Manage distribution of secure updates to edge devices
Policy Contracts	Besu	Define and enforce ABAC policies using on-chain rules
Signature Verification	Besu & FPC	Validate digital signatures in data flows
Attestation Contracts	FPC	Store and validate device attestation reports securely
TownCrier Logic Contracts	Internal (SGX)	Fetch & verify external data, invoked by Besu events
MUD Profile Contracts	Besu	Manage storage of MUD Profile

4.2 REWIRE Smart Contracts for (Application) Data Management - The TownCrier Case

4.2.1 Smart Contract Categories

The REWIRE Town Crier platform incorporates five distinct categories of smart contracts, each engineered to address specific aspects of the system's security and functionality requirements. These categories work in concert to create a comprehensive and secure blockchain-based infrastructure.

4.2.1.1 Town Crier Contracts

Town Crier contracts constitute the core oracle infrastructure of REWIRE, serving as a trusted bridge between the blockchain and external data sources. The contract implements a sophisticated request management system through a structured Request data type, handling requester information, fees, callback addresses, and parameter hashes. The contract features advanced gas price management with configurable minimum fees and cancellation charges, ensuring efficient resource utilization. It incorporates a robust upgrade mechanism with killswitch functionality for emergency scenarios, providing system

resilience. The implementation includes a comprehensive event logging system for request and delivery tracking, along with secure fee handling and distribution mechanisms to maintain system integrity.

4.2.1.2 Application Contracts

Application contracts serve as the interface layer between users and the Town Crier infrastructure. These contracts provide direct integration with Town Crier for data requests, implementing callback functionality for handling responses. They manage fees with predefined gas requirements and offer request tracking and cancellation capabilities, ensuring a seamless user experience while maintaining system efficiency.

4.2.1.3 Attestation Contracts

Attestation contracts extend the basic application functionality with specific focus on security and user management. These contracts implement a comprehensive user registration and authentication system, integrating with the DataQuerying contract for response storage. They handle secure attestation data through FPC and maintain strict access control mechanisms for registered users, ensuring data privacy and system security.

4.2.1.4 Data Querying Contracts

The DataQuerying contract implements a comprehensive data management system utilizing the ResponseData type for structured response data storage. It maintains user request tracking and history, implements contract authorization mechanisms, and provides extensive event logging for data storage operations, ensuring data integrity and traceability throughout the system.

4.2.1.5 Policy Enforcement Contracts

Policy Enforcement contracts implement device management and update distribution mechanisms. These contracts handle device authorization and policy management, process secure update requests, maintain policy data storage and retrieval systems, and integrate with Town Crier for update delivery, ensuring consistent and secure policy enforcement across the platform.

4.2.1.6 MUD Profile Contracts

MUD Profile Contracts manage the storage and retrieval of Manufacturer Usage Description (MUD) profiles associated with specific device types. Deployed on the Besu ledger, they support profile creation, updates, and lookups. These contracts ensure integrity through strict access control and integrate with SCB for signature verification and policy-based authorization. Event logging mechanisms enable auditability of all profile-related operations, supporting secure and consistent enforcement of network behavior rules for edge devices.

4.2.2 Smart Contract Implementation

The implementation follows a consistent pattern across all contracts, with two primary function types:

4.2.2.1 External Functions (E)

External functions provide the primary interface for contract interaction. These functions implement request handling with comprehensive parameter validation, ensuring data integrity at the entry point. They process responses with robust error handling mechanisms and manage data storage and retrieval operations. Additionally, they emit events for tracking and monitoring purposes, maintaining system transparency and auditability.

4.2.2.2 Update Functions (U)

Update functions manage administrative operations within the system. These functions handle contract upgrades and version management, configure system parameters, implement emergency controls and killswitch functionality, and maintain access control and authorization management, ensuring system adaptability and security.

4.2.3 Technical Implementation Details

The smart contracts implement sophisticated security measures and operational features:

Gas Management All contracts implement careful gas management through predefined gas price constants (GAS_PRICE = 5 * 10^10), minimum fee requirements (MIN_GAS = 30000 + 20000), and cancellation fee handling (CANCELLATION_FEE = 25000 * GAS_PRICE), ensuring efficient resource utilization and preventing system abuse.

Event System The contracts implement comprehensive event logging across all operations, including request and response tracking, policy and update management, user registration and authorization, and error and status reporting, providing complete system transparency and auditability.

Security Features The contracts implement multiple security layers through access control modifiers (onlyOwner, onlyAuthorizedDevice), request validation and parameter verification, secure fee handling and refund mechanisms, and state consistency checks, ensuring robust system security and reliability. All functions detailed in Table 4.3 are implemented with these security considerations and operational requirements in mind, ensuring robust and secure operation of the REWIRE ecosystem.

Table 4.2: Smart Contract Functions in REWIRE - Town Crier

Type	Function Interface	Description	Release	Accountable	Contract
E	**constructor()**	Initializes the core Town Crier contract with default parameters, establishes request tracking system and sets initial operational states	1.0	Town Crier System	TownCrier
U	**upgrade()**	Enables contract upgrade mechanism for future improvements and bug fixes by migrating to a new contract address	1.0	Town Crier Admin	TownCrier
U	**reset()**	Configures system parameters including gas price, minimum required gas, and request cancellation fees	1.0	Town Crier Admin	TownCrier
U	**suspend()**	Emergency function to pause all contract operations in case of detected vulnerabilities or system maintenance	1.0	Town Crier Admin	TownCrier
U	**restart()**	Resumes normal contract operations after suspension, reenabling all system functionalities	1.0	Town Crier Admin	TownCrier
U	**withdraw()**	Securely transfers accumulated fees and balances to authorized admin wallet	1.0	Town Crier Admin	TownCrier
E	**request()**	Creates new data request with specified parameters, callback information and custom payload for external data retrieval	1.0	User Application	TownCrier

Type	Smart Contract Function	Description	Release	Accountable	Contract
E	**deliver()**	Processes and delivers retrieved external data to requesting contract with verification and error handling	1.0	Town Crier System	TownCrier
U	**cancel()**	Allows users to cancel pending requests and receive refund of unused gas fees	1.0	User Application	TownCrier
E	**constructor()**	Sets up application contract with required Town Crier and data querying contract references	1.0	System	Application
E	**request()**	Initiates data request through Town Crier with application-specific parameters	1.0	User	Application
E	**response()**	Callback function to handle and process received data from Town Crier	1.0	System	Application
U	**cancel()**	Cancels pending application requests with proper cleanup	1.0	User	Application
E	**constructor()**	Initializes attestation contract with necessary dependencies, security parameters, and establishes connection with Town Crier for verification process	1.0	System	Attestation
U	**registerUser()**	Adds new user to the attestation system with proper verification, including identity validation and Town Crier attestation checks	1.0	Admin	Attestation
U	**unregisterUser()**	Removes user access and cleans up associated attestation data, including revocation of Town Crier verification status	1.0	Admin	Attestation
E	**isRegisteredUser()**	Verifies user registration status and access rights through Town Crier's verification system	1.0	System	Attestation
E	**getAttestationData()**	Retrieves and verifies attestation data for specific request types, including Town Crier's verification results	1.0	User	Attestation
E	**storeAttestationFPC()**	Securely stores and processes FPC-related attestation data with integrity checks and Town Crier verification integration	1.0	System	Attestation
U	**cancel()**	Terminates ongoing attestation requests with proper state cleanup and Town Crier verification cancellation	1.0	User	Attestation
E	**constructor()**	Establishes data querying contract with Town Crier integration	1.0	System	DataQuerying
U	**authorizeContract()**	Grants data access permissions to verified smart contracts	1.0	Admin	DataQuerying
E	**storeResponse()**	Securely stores and indexes response data with requester verification	1.0	System	DataQuerying
E	**getResponse()**	Retrieves stored response data with access control checks	1.0	User	DataQuerying
E	**getUserRequests()**	Compiles and returns all request history for specific user	1.0	User	DataQuerying

Type	Smart Contract Function	Description	Release	Accountable	Contract
E	**constructor()**	Initializes policy enforcement system with required contract dependencies	1.0	System	PolicyEnforcement
U	**authorizeDevice()**	Registers device for secure update distribution with proper verification	1.0	Admin	PolicyEnforcement
U	**deauthorizeDevice()**	Removes device from authorized update recipients list	1.0	Admin	PolicyEnforcement
E	**isDeviceAuthorized()**	Checks device authorization status for update eligibility	1.0	System	PolicyEnforcement
E	**CreatePolicy()**	Establishes update distribution policies with security rules	2.0	Admin	PolicyEnforcement
E	**getPolicy()**	Retrieves active policy rules for specific device	1.0	User	PolicyEnforcement
E	**requestUpdate()**	Initiates secure software/firmware update request with policy validation	1.0	User	PolicyEnforcement
E	**storePolicyUpdate()**	Processes and stores received update data with integrity verification	1.0	System	PolicyEnforcement
U	**cancel()**	Cancels pending update request with proper state cleanup	1.0	User	PolicyEnforcement
E	**GetApplicationData()**	Gets Application data and executes data filtering, data veracity. In second step uses the functions to create the pointers and store the data to off chain StoreOfchainDataAndDBPointer()	Deprecated	System	Application
U	**validateDataVeracity()**	Validates the integrity and authenticity of the data provided by devices or external sources based on the associated verifiable presentation	Deprecated	System	Application
U	**validateDataFiltering()**	Validates the integrity and authenticity of the data provided by devices or external sources based on Application related Data	Deprecated	System	Application
E	**AuthenticationAccesscontrol()**	VPs and VCs, with specific CA have access to specific channel and with several list of attributes give permissions to read specific data, for user and device	Deprecated	System	Application
U	**verifyExternalEvent()**	Verifies the occurrence and authenticity of an external event against trusted sources	Deprecated	System	TownCrier
U	**AddExternalSource()**	Adds a new external data source for the oracle to query	Deprecated	Admin	TownCrier
U	**authenticateUser()**	Authenticates a user's identity before allowing access to sensitive functions	Deprecated	System	TownCrier
E	**EmitEvent()**	Emits a blockchain event in response to specific triggers or conditions	Deprecated	System	TownCrier

Table 4.3: Smart Contract Functions in REWIRE - MUD Profile Contracts

Type	Function Interface	Description	Release	Accountable	Contract
E	**createOrUpdateMUDProfile()**	Creates a new or updates an existing MUD profile for a given device type. Includes configuration hash and vulnerability metadata.	1.0	MUD Profile Server	MUD Profile
E	**getMUDProfile()**	Retrieves the current MUD profile for a specified device type.	1.0	Edge device	MUD Profile
E	**getMUDProfileHistory()**	Returns historical versions of MUD profiles for audit and verification purposes.	1.0	Edge device	MUD Profile

Note on Deprecated Functions: The following functions from D5.1 have been deprecated in D6.1 due to architectural changes and implementation decisions:

- **GetApplicationData()**, **validateDataVeracity()**, **validateDataFiltering()**, and **Authentication Accesscontrol()**: These functions were replaced by a more streamlined data handling approach in the Application contract, where data validation and access control are now integrated into the core request/response flow.
- **verifyExternalEvent()**, **AddExternalSource()**, **authenticateUser()**, and **EmitEvent()**: These functions were removed from the TownCrier contract as their functionality is now handled through the enhanced policy enforcement system and integrated security mechanisms.

4.2.4 Smart Contract Interactions

The REWIRE system implements a sophisticated interaction flow between its various smart contracts, enabling secure and efficient data management across the platform. This interaction pattern can be categorized into three main flows: Application Contract Flow, Attestation Contract Flow, and Data Query Flow.

4.2.4.1 Application Contract Flow

The Application Contract Flow initiates with a request call to the Town Crier contract, containing essential parameters including requestType, callbackAddr, callbackFID, timestamp, and requestData. Upon receiving this request, the Town Crier contract generates and returns a unique requestId to the application. The Town Crier then monitors and fetches data from external data sources, implementing a secure bridge between the blockchain and external systems. Once the external data is received, the Town Crier processes this raw data and initiates a storeResponse operation to the DataQuerying contract, providing the requestId, requester address, error status, and response data. Upon successful storage confirmation, the Town Crier contract triggers a response callback to the original application, completing the request cycle with appropriate error handling and data validation.

4.2.4.2 Attestation Contract Flow

The Attestation Contract Flow follows a similar pattern but with specialized handling for attestation data. The process begins with a getAttestationData call to the Town Crier contract, specifying the requestType and requestData parameters. After receiving the requestId, the Town Crier contract fetches the required attestation data from external sources. The retrieved data undergoes secure processing before being stored through the storeResponse mechanism in the DataQuerying contract. The final step involves the

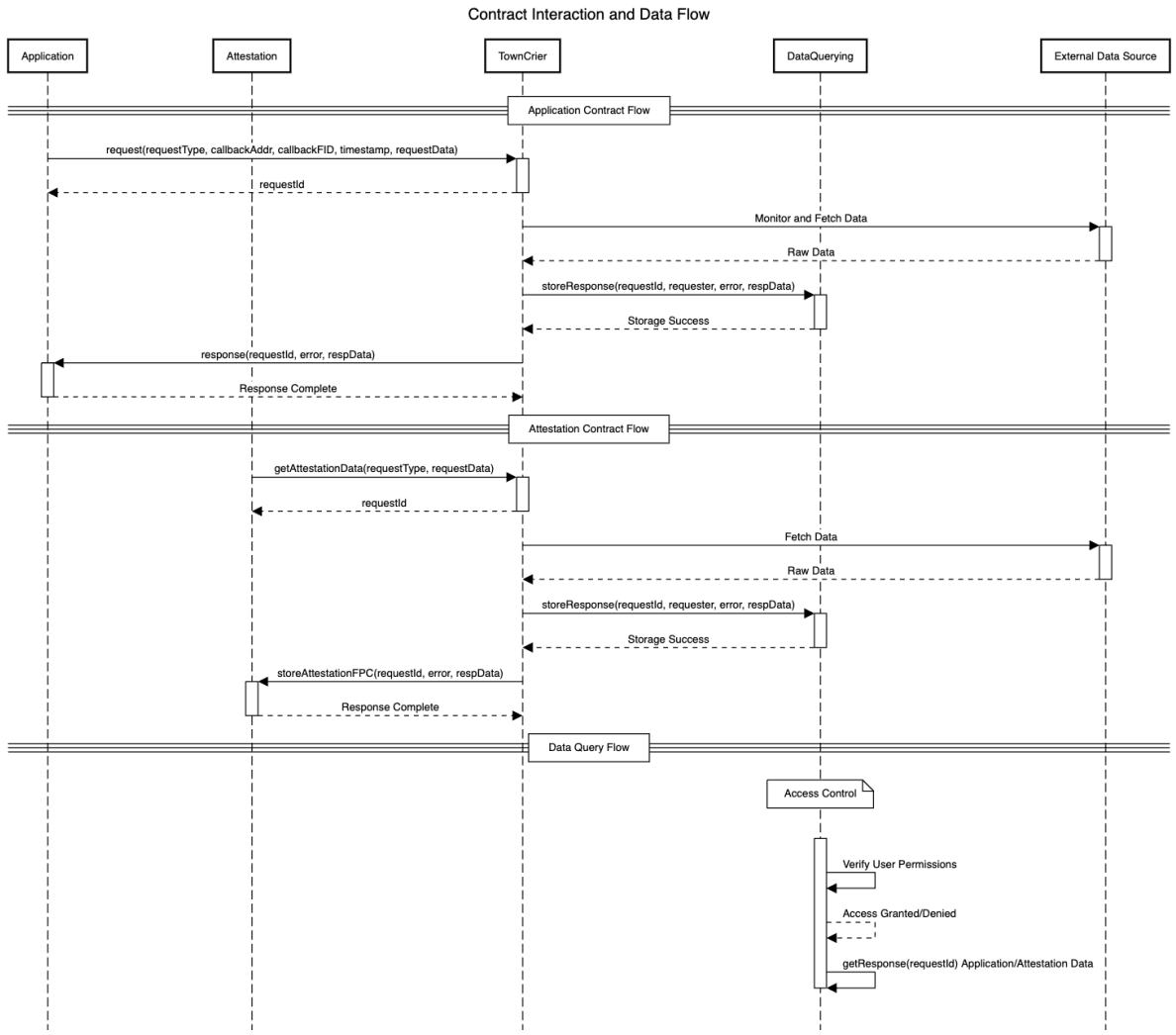

Figure 4.1: Sequence diagram illustrating the interaction flows between TC smart contracts

Town Crier contract calling storeAttestationFPC on the Attestation contract, which handles the specialized storage and verification of attestation-specific data, implementing additional security measures for attestation integrity.

4.2.4.3 Data Query Flow

The Data Query Flow implements a secure access control mechanism for retrieving stored data. This flow begins at the DataQuerying contract, which implements sophisticated access control checks before processing any data retrieval requests. When a request is received, the contract first verifies user permissions through its access control mechanism. Based on the verification result, access is either granted or denied. For authorized requests, the contract retrieves and returns the requested Application or Attestation data, maintaining strict security protocols throughout the process. This flow ensures that sensitive data is only accessible to properly authenticated and authorized entities within the system.

4.2.4.4 Cross-Contract Communication

The interaction between contracts is carefully orchestrated to maintain security and data integrity. Each contract implements event emission for significant state changes, enabling transparent tracking of cross-contract operations. The communication patterns are designed to minimize gas costs while ensuring reliable data transmission between contracts. Error handling mechanisms are implemented at each interaction point, ensuring robust operation even in case of unexpected conditions or failed operations.

These interaction patterns ensure secure, efficient, and reliable operation of the REWIRE ecosystem, enabling complex data management and attestation processes while maintaining strict security requirements.

4.2.4.5 MUD Profile Contract Flow

The MUD Profile Contract Flow handles the lifecycle of Manufacturer Usage Description (MUD) profiles within the REWIRE blockchain infrastructure, ensuring integrity, authenticity, and controlled dissemination of security configurations for IoT edge devices.

The process begins with a createOrUpdateMUDProfile transaction initiated by the MUD Profile Server. This transaction includes the device type identifier, the updated configuration hash, and optionally any associated threat or vulnerability metadata. The MUD Profile contract verifies the sender's authorization before accepting and persisting the profile update to the Besu ledger.

Upon successful storage, the MUD Profile contract emits a MUDProfileUpdated event, which the SCB subscribes to. The SCB, acting as the orchestrator, then issues a notification to all edge devices registered for that device type, informing them of the availability of a new or modified MUD profile. When a device receives this notification, it initiates a secure query through the SCB to retrieve its corresponding profile. This query triggers a getMUDProfile or getMUDProfileHistory call on the MUD Profile contract, which responds with the latest or historical profile data, depending on the query type. Throughout this interaction, access control enforcement and signature verification steps are performed by the SCB to ensure only authorized devices can access the profile data.

This flow ensures timely propagation of updated security policies, while maintaining accountability and auditability via logged transactions and structured event emissions on-chain.

4.2.5 Device Secure Management through MUD Profiles

The REWIRE platform introduces secure device management capabilities through the structured use of Manufacturer Usage Description (MUD) profiles, which are anchored on the Besu blockchain via dedicated MUD Profile smart contracts. These profiles serve as immutable and verifiable policy artifacts that define the intended behavior, configuration, and security posture of edge devices.

Whenever new vulnerabilities are discovered or updated threat intelligence is received, the MUD Profile Server computes a new configuration hash reflecting the revised security posture. This may include updated network fingerprints, revised access policies, and extended lists of threat identifiers associated with specific device types. The updated MUD profile is submitted via a createOrUpdateMUDProfile call, securely stored on-chain, and disseminated through SCB-mediated notifications to the corresponding edge devices.

By anchoring these updates in the REWIRE blockchain infrastructure, the system ensures:

- Verifiable provenance of security updates.
- Immutable history of device policy evolution.
- Real-time dissemination to affected devices.

This approach strengthens the platform's trust and accountability model, enabling resilient and transparent management of connected devices through policy-driven, cryptographically enforced control.

4.3 Trust Assessment & Auditability - The FPC Case

In REWIRE's blockchain infrastructure, the Fabric Private Chaincode (FPC) plays a critical role in ensuring trust, confidentiality, and auditability of sensitive security data such as attestation reports. As the privacy-preserving ledger component built on top of Hyperledger Fabric and Intel SGX, FPC enables on-chain computation with strong confidentiality guarantees. Within this environment, REWIRE has developed a suite of smart contracts to manage and validate the lifecycle of attestation data—core to the trust assessment pipeline.

These contracts are designed to not only store and retrieve data, but also enforce strict verification mechanisms to assess the trustworthiness of incoming inputs and the identities providing them. Furthermore, the latest version of REWIRE introduces **on-chain cryptographic verification** of signatures using ECDSA, bringing the system closer to decentralized, standards-aligned trust frameworks.

4.3.1 Smart Contract Functions

The smart contracts deployed within the FPC enclave implement key logic for managing and evaluating attestation data. These include:

- **storeAttestationReport(device_id, payload, vp, signature)**: This function accepts an attestation report from a device, verifies its authenticity via cryptographic means and policy validation, and securely stores it on the FPC ledger.
- **getAttestationReport(device_id)**: Retrieves the latest attestation report for a specified device. This supports downstream components (such as SW/FW analysis modules) in assessing device trustworthiness.
- **getAllAttestationReports(device_id)**: Provides the full history of attestation reports for a device, supporting auditability, historical data analysis, and long-term monitoring.
- **verifySignature(payload, signature, public_key)**: Validates that the payload's digital signature is correct and matches the provided public key using an on-chain ECDSA verification process executed within the SGX enclave.

All of these functions are executed inside the SGX enclave, and are mediated through the SCB, which enforces ABAC policies and ensures authenticated interaction. Table 4.4 summarises these smart contract functions.

Table 4.4: Overview of FPC Smart Contracts

Function	Description	Input	Output	Security Enforcement
storeAttestationReport	Stores verified attestation report	device_id, payload, vp, signature	Success/ Failure flag	ECDSA signature & Attribute verification
getAttestationReport	Retrieves latest attestation report	device_id	Report payload	SCB-mediated ABAC check
getAllAttestationReports	Retrieves all reports for device	device_id	Report list	SCB-mediated ABAC check
verifySignature	Verifies ECDSA signature of payload	payload, signature, public_key	True/False	Executed securely on FPC's chain

4.3.2 Technical Implementation Details & Specifications

The smart contracts in FPC are developed in Go and adhere to the secure development standards provided by the FPC SDK. Each contract is compiled to run inside Intel SGX enclaves, ensuring that even sensitive cryptographic operations remain confidential and protected against external tampering.

A key highlight in this version of REWIRE is the integrated signature verification mechanism. The verifySignature function performs ECDSA verification within the enclave, ensuring that data received on-chain is cryptographically valid. This mechanism aligns with W3C-SSI principles and provides a high degree of trust in data origin and integrity.

Additionally, the smart contracts support parsing and evaluating Verifiable Presentations (VPs), which are attached to the payloads submitted by devices. These VPs are structured according to W3C standards and include identity attributes used to validate that the device has the necessary authorization. This VP is evaluated against embedded policy rules.

Once verified, the data is stored within the FPC's private collections using secure ledger mechanisms that guarantee confidentiality and auditability. All access to these contracts is controlled via the SCB, which ensures that only authenticated and authorized users or components can invoke FPC functionality, in accordance with predefined ABAC policies.

Table 4.5: Internal Workflow for storeAttestationReport

Step	Operation	Component
1	Validate digital signature	FPC Enclave (verifySignature)
2	Parse and validate Verifiable Presentation	FPC Enclave
3	Check against attestation policy	PC Enclave
4	Store validated report in the ledger	FPC Ledger
5	Return transaction status to SCB	SCB

4.3.3 Implementation Details of the Verification Process

One of the significant enhancements introduced in the latest iteration of REWIRE's Fabric Private Chaincode (FPC) implementation is the inclusion of on-chain signature verification. This upgrade moves trust validation deeper into the confidential computing layer, ensuring that attestation data is cryptographically validated before being persisted on the ledger.

In this implementation, the signature verification process follows an **ECDSA (Elliptic Curve Digital Signature Algorithm)** scheme, which is widely adopted in secure decentralized systems. The device submitting the attestation report signs the payload off-chain using its private key. The signed report is submitted alongside the public key (derived from the device's verifiable credential) and the Verifiable Presentation (VP). Inside the FPC enclave, the smart contract calls the verifySignature() function, which:

1. Parses the payload and ECDSA signature.
2. Uses the provided public key to validate the digital signature against the submitted data.
   (a) If valid, proceeds with policy checks on the attached VP.
   (b) If any check fails, the transaction is rejected, ensuring no unverified data ever enters the ledger.

This flow is tightly integrated with the W3C Self-Sovereign Identity (SSI) framework. The VP, which encapsulates identity attributes, is formatted according to W3C standards. During processing, the FPC smart contract evaluates whether these attributes meet REWIRE's defined policies (e.g., ensuring the report was generated by a device within a trusted domain or of a specific type).

By performing this validation inside the SGX enclave, REWIRE significantly reduces the risk of malicious data injection, external manipulation, or incorrect author attribution. It also ensures compliance with zero-trust principles and decentralization goals, making the trust model both scalable and formally verifiable.

4.3.4 SSI-based Identity Management & Access Control

REWIRE's access control and identity verification mechanisms are rooted in the **Self-Sovereign Identity (SSI)** paradigm, which gives full control to devices and users over their digital identities. This identity model fits naturally with REWIRE's privacy-preserving design goals, enabling decentralized trust enforcement across all smart contract operations.

Each device onboarded into REWIRE is issued a **Verifiable Credential (VC)** by a trusted VC manager. These credentials encode the device's attributes, such as hardware type, manufacturer, software stack, trust domain, and certification level. When interacting with FPC contracts, devices must present **Verifiable Presentations (VPs)** generated from these VCs. These VPs are signed proofs that contain selected attributes required for the interaction at hand (e.g., submitting or querying attestation reports).

Access control is enforced through **ABAC** policies that are embedded within the SCB and respected by the FPC enclave. These policies ensure that:

- Only authorized devices or users with specific attributes can store or retrieve data.
- Query operations are logged and tied to VP-based identities.
- Privacy and minimal disclosure are maintained, as VPs allow selective attribute sharing.

By decentralizing identity control and enforcing rich, attribute-driven access logic, REWIRE achieves granular and flexible data governance. This is particularly crucial in scenarios involving sensitive attestation information, software integrity verification, or regulatory compliance, where precise authorization checks and non-repudiation are mandatory.

Furthermore, this approach makes the FPC layer not just a storage layer, but a **trust-aware computation environment**, tightly aligned with REWIRE's broader vision of secure, auditable, and privacy-preserving data flows.

Chapter 5

Security Context Broker Functional Specification and Technical Implementation

The **Security Context Broker (SCB)** is a foundational component within the REWIRE blockchain infrastructure, acting as the unified coordination and execution layer that bridges the various technologies integrated in the system. Built as a **Node.js server**, the SCB is equipped with all the necessary SDKs and libraries for seamless interaction with REWIRE's heterogeneous stack, **Hyperledger Fabric (FPC), Ethereum-based Besu blockchain, TownCrier secure oracle, and the off-chain data store (ELK stack)**.

The SCB mediates secure and structured interactions across the infrastructure, exposing high-level interfaces to other internal REWIRE components while encapsulating and abstracting away complex low-level logic. It enforces **ABAC**, handles **identity-based trust validation**, and facilitates **synchronous orchestration** of multi-layered transactions that span from the blockchain layer to secure computation enclaves.

5.1 SCB Acts as a Transaction Bridge

The Security Context Broker (SCB) functions as the core transactional hub within the REWIRE blockchain infrastructure, acting as a **trusted mediator** and **secure coordination layer** for all interactions between blockchain components and external actors. Its primary responsibility is to **orchestrate and enforce secure data-sharing workflows**, serving as the **entry point for initiating blockchain-related transactions** and controlling access to verifiable data across the REWIRE ecosystem.

SCB is responsible for bridging external requests (originating from REWIRE modules such as the AI-based Misbehavior Detection Engine, Risk Assessment, or Firmware Distribution Service) to the appropriate blockchain backend components, including **Besu**, **TownCrier, Fabric Private Chaincode (FPC)**, and the **off-chain data storage layer**. This is achieved through synchronous and asynchronous message passing, coupled with **fine-grained access control enforcement** based on **ABAC** policies and **VPs**. Implemented as a **Node.js server**, the SCB integrates all the necessary SDKs and cryptographic libraries to securely interact with REWIRE's diverse technology stack. These SDKs abstract the underlying complexity of each component—whether it's invoking chaincode on FPC, triggering event publication on Besu, or managing off-chain pointer resolution—allowing SCB to act as a **unified abstraction layer** for secure transaction logic.

A key functional aspect of SCB as a transaction bridge is its **event-driven architecture**. For example, when a device finalizes an attestation report, SCB generates a corresponding blockchain event on Besu. This event is picked up by the TownCrier secure oracle, which then initiates the secure data-fetching process. Similarly, when large data sets are fetched or stored, SCB ensures that decisions are made regarding whether to route them through Besu or off-chain storage, enforcing thresholds and trust policies in real time.

Moreover, the SCB plays a critical role in **auditability** and **compliance enforcement** by logging every invocation, transaction, and access request, thus preserving a verifiable trace of system behavior. In essence, the SCB doesn't just act as a gateway; it **actively governs** the secure, verifiable, and policy-compliant flow of data throughout the REWIRE framework. It is purposefully centralized to reduce the attack surface, simplify trust boundaries, and ensure full accountability in cross-ledger and cross-component operations, while still enabling decentralized data provenance via blockchain infrastructure.

5.2 Interface Specification & Implementation

The SCB exposes a focused set of high-level interfaces intended for use exclusively within the REWIRE infrastructure. These interfaces enable trusted REWIRE components to securely interact with blockchain-backed data—ensuring compliance with access control policies and maintaining system integrity through authenticated, verifiable interactions. The Table 5.1 below presents the core interfaces exposed by the SCB.

Table 5.1: Externally Accessible SCB Interfaces

Interface ID	Name	Description	Invocation	Notes
SCB.EX.1	Query Application Data	Allows authorized REWIRE services to retrieve validated application data via Besu/TownCrier.	HTTPS + ABAC	Requires Verifiable Presentation; internally mapped to Besu queries via TownCrier oracle.
SCB.EX.2a	Query Latest Attestation Reports	Enables secure access to the latest attestation reports stored in FPC, based on requester's access policies.	HTTPS + ABAC	Signature and attribute checks are enforced before retrieval.
SCB.EX.2b	Query All Attestation Reports	Enables secure access to all attestation reports stored in FPC, based on requester's access policies.	HTTPS + ABAC	Signature and attribute checks are enforced before retrieval.
SCB.EX.3a	Store MUD Profiles	Publishes Manufacturer Usage Description (MUD) profiles for registered IoT devices.	HTTPS	Interfaced with internal SCB device registry; used for behavior profiling and risk checks.
SCB.EX.3b	Query MUD Profiles	Retrieves Manufacturer Usage Description (MUD) profiles for registered IoT devices.	HTTPS	Interfaced with internal SCB device registry; used for behavior profiling and risk checks.
SCB.EX.4	Query SW/FW Update	Retrieves SW/FW Update for the registered IoT devices.	HTTPS + ABAC	Interfaced with internal SCB device registry; used for SW/FW Update.

All interfaces are RESTful and implemented in **Node.js**, making use of standard JSON schemas. Before execution, the SCB validates each request using aligned **W3C Verifiable Presentations**, performing cryptographic checks and attribute evaluation based on policy rules. While these are the public-facing endpoints within REWIRE, the SCB also includes a comprehensive set of internal, event-triggered interfaces that orchestrate the secure data flows across components like Besu, FPC, TownCrier, and the off-chain storage. These internal interfaces are not exposed externally and are executed automatically based on pre-defined event triggers, ensuring seamless operation across the blockchain infrastructure without manual intervention. Together, this architecture enables the SCB to function as both a secure policy enforcer and a reliable dispatcher for inter-component logic across REWIRE.

Chapter 6

REWIRE Blockchain Infrastructure Performance Evaluation

This chapter, as part of REWIRE's final system validation, presents a **comprehensive performance evaluation of the REWIRE Blockchain infrastructure**. This effort builds upon the benchmarking activities presented in Deliverable D6.1 [6], extending the scope to end-to-end, system-level measurements that capture the interconnected execution paths of key REWIRE components under realistic operational workloads. While Deliverable D6.1 focused on micro-benchmarking individual functions, such as enclave operations and contract invocations, the current analysis shifts attention to the **overall responsiveness, scalability, and feasibility** of the fully deployed and **integrated system**, thereby validating its **suitability for production-grade use**.

The evaluation encompasses the full REWIRE Blockchain Infrastructure, including data consumption, attestation, SW/FW patching, policy distribution, and application data queries. This system-wide perspective enables the identification of latency contributors, performance bottlenecks, and inter-component dependencies across multiple trust-critical scenarios. Special attention is given to the **Secure Oracle Layer**, an architectural element in REWIRE, including the TC, and the FPC two TEE-based secure oracles responsible for verifiable data acquisition, cryptographic attestation, and confidential on-chain operations. It is important to distinguish that the SCB and Hyperledger Besu—although crucial, are not part of the Secure Oracle Layer. Rather, SCB acts as a policy-aware trust mediator, enforcing verifiable identity and ABAC policies, while Besu serves as a synchronization and anchoring ledger, coordinating the data flow between TC and FPC. Together, these components contribute to the overall data veracity and transactional integrity across REWIRE's workflows.

The **performance results** presented in this chapter not only reveal the latency and throughput characteristics of the individual components, but also **expose broader architectural and research challenges**. These include the **orchestration** of secure enclaves, **cross-oracle synchronization**, and interoperation under **real-world conditions**. This analysis goes beyond REWIRE's benchmarking, it serves as a critical design validation, highlighting: (a) the need for systematic optimization in multi-enclave workflows, (b) the current barriers to direct interoperability between secure oracles due to heterogeneous TEE stacks and enclave APIs and (c) the broader architectural implications for integrating multi-oracle environments in a verifiable and confidential way. REWIRE serves as a foundational step toward building **interoperable, multi-oracle secure infrastructures**. It contributes not only a functional implementation, but also an informed research agenda pointing to future exploration of TEE-enabled roll ups, Phala Network, Intel TDX, and other privacy-preserving blockchain oracle solutions. Ultimately, the evaluation in this chapter affirms the maturity, extensibility, and mission-readiness of REWIRE's Blockchain Infrastructure.

Finally, moving toward a **Verifiable Network of Trust**, it should be emphasized that this system-level evaluation is not an isolated exercise, but rather the culmination of the technical trajectory laid out throughout the REWIRE project. It demonstrates the feasibility of Blockchain-integrated security mechanisms capable of enabling a Verifiable Network of Trust, one in which Blockchain peers are not merely storage nodes,

but active verifiers of data provenance, authenticity, and policy compliance. In this capacity, REWIRE advances the state of the art by transforming smart contract infrastructures into **end-to-end verifiable, privacy-preserving ecosystems**, suitable for deployment in mission-critical and safety-critical domains. The resulting infrastructure offers a strong foundation for trustworthy distributed applications that demand transparency, control, and cryptographic trust at scale.

Evaluation Objectives and Core Functionalities Performance evaluation was anchored in two core functional domains central to REWIRE's vision for secure device lifecycle management that were not defined and evaluated in previous deliverables. First, the onboarding and personalization of security policies, which relies on extracting configuration metadata, software stack, and known threats from the Manufacturer Usage Description (MUD) profiles. These profiles serve as a basis for instantiating key usage restrictions and secure communication policies during a device's initial enrolment into the system. Second, the one-to-many SW/FW Update distribution process, in which edge devices securely receive authenticated patches through the Besu ledger, while enforcement policies and attestation proofs ensure that only authorized devices complete the update. Although these functionalities were previously mentioned in early deliverables, they were not evaluated until this stage. Thus, the present chapter not only assesses general data consumption and querying workflows, but also provides the first holistic evaluation of device onboarding and SW update distribution, measuring their execution under system-level constraints.

Use Case Alignment and System Emulation In order to emulate production-level behaviors, the test environment incorporated input parameters and workload modalities derived from REWIRE's two representative use cases. The **Smart Satellite Use Case** provided the basis for evaluating the consumption of **application-related data** and **MUD-based configuration flows**. This scenario emphasized periodic data push patterns and varied payload sizes, reflecting the telemetry and security policy update mechanisms of intelligent, orbiting edge devices. In parallel, the **Smart City Use Case** served as the basis for ingesting **attestation datagrams** and also modeling large-scale, concurrent SW update dissemination. The system was subjected to **one-to-many updates** targeting a number of devices, thus enabling a stress-test of the SCB's orchestration and Besu's consensus throughput.

Measuring Security Overheads The evaluation also considers the performance impact of the trust mechanisms that underpin REWIRE's secure data exchange and privacy-preserving infrastructure. While not quantifying trust itself, this analysis captures the latency introduced by components that enable secure and authenticated handling of trust-related evidence. Specifically, it tracks the overhead introduced by signature verification, SGX enclave processing, policy enforcement through ABAC, and on-chain anchoring of evidence and datagrams. These operations are essential for supporting dynamic, authorized trust assessments within REWIRE, ensuring that trust data is processed and accessed only by verified actors and according to defined policies. The measurements thus illustrate how REWIRE balances strong security guarantees with real-time performance requirements, particularly in latency-sensitive edge computing scenarios.

Confidentiality and Private Datagrams While blockchain infrastructures are widely recognized for their strengths in ensuring integrity and auditability, their support for confidentiality, particularly in the context of metadata and transaction input protection, remains an area of ongoing evolution. In REWIRE, confidentiality is not treated in isolation but rather as a key enabler of data veracity, ensuring that both the authenticity and privacy of sensitive interactions are preserved throughout the data pipeline. To this end, REWIRE introduces confidential datagram request flows between SCB and secure enclaves, shielding identity-revealing or context-sensitive requests from public exposure. These encrypted and authenticated datagrams form the basis of trust-preserving interactions between off-chain sources and the Blockchain layer, particularly in the TC Oracle's role as a secure consumption gateway. Expanding on this model,

REWIRE employs custom datagrams—data packets that are adaptively configured based on pre-defined fetching intervals and payload size, tailored to application-specific trust and performance requirements. Throughout the evaluation presented in this chapter, all datagram exchanges were constrained to payloads under 20 KB, ensuring that data could be fully anchored on-chain without resorting to off-chain fallback storage. This threshold reflects realistic operational constraints and maximizes traceability and auditability while capturing the latency and orchestration costs associated with private, verifiable, and context-aware data exchange.

6.1 Experimental Setup & Evaluation Properties of Interest

Evaluation Context and Motivation Performance measurements were conducted in a **virtualized, TDX-compatible server-based testbed**, offering a controlled and realistic environment that emulates the expected operational characteristics of the REWIRE Blockchain Infrastructure. This setup enabled the observation of system behaviors under **end-to-end** conditions and integrated deployments across the full stack of trusted components. As aforementioned, the benchmarking activities revolved around the two representative and safety-critical domains drawn from REWIRE's envisioned use cases: **smart satellites** and **smart cities**. These domains were selected because the reliability, security, and verifiability of data management are essential to their operation. In such environments, the performance of Blockchain-assisted operations is not merely a technical consideration, it directly impacts mission assurance and service reliability.

Application Domain Target

- **Smart Satellites – Confidential Telemetry Management**: In the satellite domain, data flows are characterized by high-frequency transmissions, heterogeneous sensor sources, and dynamically varying payload sizes. To properly reflect these characteristics, REWIRE employed synthetic data generation enriched by patterns derived from real-world telemetry datasets (as discussed in [5]). This scenario also emphasized the confidentiality and integrity of telemetry ingestion: sensitive operational data must remain private, while the verifiability of the data's origin and content is paramount. As such, the experimental flow incorporated custom and private datagram mechanisms, processed securely via the SCB and TC Oracle, to emulate enforcement of ABAC and the traceability of smart contract execution.
- **Smart Cities – Secure Policy Enforcement at Scale**: In contrast, the smart city domain emphasized device-wide trust management and policy compliance in dynamic large-scale IoT environments. The dominant scenario in this domain was the evaluation of attestation evidence consumption, which serves as the foundation for assessing and maintaining the trustworthiness of connected devices over time. This scenario simulated the secure storage, retrieval, and policy-based querying of device attestation reports, leveraging the TC Oracle, Hyperledger Besu, and the FPC enclave to ensure confidentiality, integrity, and fine-grained access control across the data lifecycle. Crucially, this attestation flow is not isolated, it acts as a precursor to policy-driven actions. For example, when attestation evidence indicates a degraded or altered trustworthiness state for a device, this can trigger a policy-based response, such as initiating a secure SW/FW Update to restore compliant behavior. This leads directly to the second key scenario, which focused on one-to-many SW/FW Update distribution, a critical operation for synchronizing policy enforcement and patch deployment across swarms of heterogeneous edge devices. The use of a distributed ledger (Hyperledger Besu) in this context was not an arbitrary design choice; it was essential to ensure replication, auditability, and compliance assurance at scale. Update propagation followed event-driven flows, policy-guarded delivery paths, and relied on anchored metadata validated within secure enclaves and notarized on-chain. In both smart city scenarios, custom datagram configurations, adaptively sized and temporally controlled, were used to emulate device-level access control policies and to

benchmark latency, scalability, and throughput in high-density, concurrent-access environments. Together, these workflows demonstrate the tight coupling between real-time trust assessment and automated remediation, showcasing how REWIRE supports secure lifecycle management across complex IoT ecosystems.

Security Properties Under Evaluation Each of the evaluated flows focused on the distinct security and performance properties that define their operational context and are provided in Table 6.1.

Table 6.1: Evaluated flows and properties

Evaluated Flow	Security & Performance Properties
Application-related Data Consumption & Querying	- Integrity via VP signature validation and on-chain anchoring in Besu - Confidentiality through custom datagrams, secure HTTPS fetch operations, and SGX-based processing, w3c aligned
Attestation-related Evidence Handling	- Cryptographic validation of device trustworthiness inside Town Crier and FPC enclaves - Traceable, policy-enforced querying of attestation histories for dynamic trust characterization
SW/FW Update Workflows	Scalability validated through high-concurrency one-to-many consumption and delivery - Policy enforcement via ABAC in SCB and smart contract-based access decisions characterization
Security Policy (MUD Profile) Distribution	- Fine-grained access control via VP verification and ABAC filtering - Scalable policy dissemination with secure retrieval through blockchain-backed processes

Classification of Operations: Online vs. Offline To obtain a comprehensive and realistic understanding of the performance of the REWIRE Blockchain Infrastructure, we classified the evaluated operations into online and offline workflows. **Online operations** refer to core, latency-sensitive functions that are executed frequently and directly affect runtime responsiveness. These include all four of the operational workflows detailed in Table 6.1: application data consumption and querying, attestation evidence handling, software/firmware update workflows, and MUD profile distribution. Performance evaluation in this chapter focuses primarily on these online operations, as they are critical to real-time service provisioning and are triggered by dynamic system events, external inputs, or time-sensitive trust decisions. **Offline operations**, by contrast, include auxiliary system setup tasks and initial trust establishment procedures, such as secure device onboarding, policy initialization, or cryptographic anchor provisioning. While are not executed regularly during runtime, these operations play a crucial role in ensuring system integrity and long-term trust guarantees. Although not the focus of the latency measurements reported here, their impact on the overall security architecture remains significant.

Device Onboarding and Policy Personalization A key architectural extension evaluated during this phase was the secure device onboarding and registration process, based on the enhanced MUD profile. REWIRE extended the standard EAP protocol to allow for the real-time parsing of configuration metadata, device threat fingerprints, and cryptographic key structures. These were used, among other purposes, as inputs to derive the necessary key restriction usage policies, particularly relevant for Blockchain-based enforcement. This derivation process was part of the broader secure domain registration workflow, which also encompassed the issuance of appropriate credentials and the formal establishment.

Infrastructure Components To support this multifaceted evaluation, the following two virtualized infrastructure components were deployed:

VM1 – Fabric Private Chaincode (FPC):

- **Purpose**: Execution of confidential chaincode and secure attestation storage.

- **Specs**: 6 CPU cores, 16 GB RAM, 100 GB disk.
- **Hardware**: Intel SGX via passthrough (/dev/sgx_enclave, /sgx_provision, /sgx_vepc).
- **SGX Configuration**: Enabled with +sgx, +sgx2, +sgxlc, and 64 MB EPC memory allocation.

VM2 – Hyperledger Besu, TownCrier, SCB Server:

- **Purpose**: Serves as the primary execution environment for Blockchain and secure oracle services.
- **Specs**: 4 CPU cores, 16 GB RAM, 200 GB disk.
- **Town Crier Oracle**: Runs within an SGX enclave to handle off-chain data acquisition and VP validation.
- **Security Context Broker (SCB)**: A Node.js-based server integrating SDKs for Besu, FPC, and ElasticSearch to coordinate queries, enforce policies, and manage external data flows.

Both enclaves, TC and FPC, leverage hardware-assisted TEEs with cryptographic acceleration capabilities, such as those provided by Intel SGX/TDX instruction sets. This hardware acceleration significantly enhances the performance of security-critical operations like ECDSA signature verification, hash computations, and secure memory handling. As a result, cryptographic operations executed inside the enclaves (e.g., VP verification, secure transaction construction) exhibit noticeably lower latency than equivalent operations handled outside the enclave (e.g., in the SCB), where such hardware-level optimizations are not available. This performance differential is reflected in multiple benchmarking results throughout this chapter and highlights the benefit of tightly isolating verifiable logic within secure enclave boundaries.

Off-Chain Storage Considerations (Elasticsearch Layer) Although this chapter focuses on evaluating REWIRE's Blockchain-enabled functional flows (onboarding, ingestion, querying, and secure updates), it is important to contextualize the role of off-chain storage in the broader system architecture. In particular, REWIRE integrates Elasticsearch as the main off-chain data store, supporting scalable and rapid access to data that exceed the on-chain size threshold (20 KB). To assess this component, REWIRE Deliverable D6.1 [6] (Section 11.4) presented an extensive performance evaluation of the Elasticsearch-based module. This included benchmarks across four dimensions in Table 6.2.

Table 6.2: Off-Chain Storage Performance

Evaluation Dimension	Observation	Performance Summary
Data Size	Upload time increased modestly with payload size.	**Upload**: 0.003s (1,000 bytes) → 0.011s (100,000 bytes) **Query**: Stable at 0.001–0.002s
Concurrency	Upload latency grew with more concurrent requests; query performance remained resilient.	**Upload**: 0.002s (1 req) → 0.022s (50 reqs) **Query**: Up to 0.010s
Batch Size	Bulk uploads are more efficient than single uploads.	Per-document upload time decreased significantly with larger batch sizes.
Query Type	Different query types had minimal effect on latency.	All types (term, match, range, wildcard): 0.002s

Overall, these results validated the robustness and scalability of REWIRE's off-chain storage design, confirming its suitability for both real-time and archival use cases.

In the current performance evaluation, all data flows were deliberately configured as datagrams with sizes below the 20 KB threshold. As a result, all records were stored directly on-chain, either via Besu

or FPC, and interactions with the off-chain storage module (Elasticsearch) were not part of the active benchmarking scope. Consequently, the performance results reported in this section reflect only the on-chain execution characteristics and do not capture the latency or behavior of off-chain storage workflows. Nonetheless, the off-chain infrastructure remains fully integrated within the REWIRE system and is operationally ready to support scenarios requiring larger payloads.

6.1.1 Evaluation Methodology

Scope of Measurement: End-to-End and Component-Level Profiling The evaluation focused on capturing end-to-end latency while also dissecting the component-wise operational profile of each functional flow. This two-fold methodology enables both holistic and fine-grained analysis of how data sharing transactions propagate across REWIRE's Secure Oracle Layer. The flows were decomposed into discrete stages, each instrumented for latency capture and synchronized across trusted components. Each transaction, from secure data ingestion to verified querying, involved coordinated activity among the SCB, TC, Besu, and FPC. In this context, the SCB serves as the secure mediator for request forwarding, policy enforcement, and data orchestration, while TC and FPC provide the SGX-secured runtime environments for cryptographic verification and attestation validation.

Measurement Approach: Hybrid Timing Strategy To capture both user-perceived latency and internal system behavior, a hybrid black-box and white-box measurement strategy was employed. From the black-box perspective, timings were recorded at the interface level, capturing the latency experienced by invoking entities such as edge devices or services during operations like data ingestion or querying. This reflects the total round-trip time for each transaction from the system's external viewpoint. Complementarily, the white-box perspective focused on capturing internal execution timings across key components of the secure oracle layer. While micro-benchmarking within SGX enclaves was covered in D6.1, here we focused on measuring the time consumed by orchestration and coordination logic across SCB, Besu, TC, and FPC. This includes delays due to ABAC enforcement, VP signature validation, smart contract preparation and submission, blockchain write latency, and enclave-protected operations within Town Crier and FPC. For example, during application data ingestion, we measured SCB's orchestration of datagram forwarding, secure data retrieval via HTTPS within the enclave, cryptographic verification, and contract anchoring on the Besu ledger. In attestation-related flows, measurements included the SCB's notification to the FPC enclave upon Besu confirmation, followed by querying, VP verification, and final on-chain storage of the attestation result. This methodology enables attribution of delay to specific functional stages, particularly those involving privacy-preserving or security-critical logic, thereby offering a comprehensive latency profile of REWIRE's secure data transaction pipeline.

REWIRE Blockchain Trusted Computing Base As part of the overall REWIRE trust model, this deliverable defines the Trusted Computing Base (TCB) specific to the deployed Blockchain Infrastructure. This TCB complements the far-edge TCB established in Deliverable D4.3, collectively safeguarding the integrity and confidentiality of operations across the full compute continuum, from edge devices to backend trust services. In the context of the REWIRE Blockchain layer, the TCB is composed of the enclaves instantiated within TEEs that underpin the core operations of the two secure oracle components: TC and FPC. These enclaves are essential to the verifiable and confidential execution of integrity-critical processes, including Verifiable Presentation (VP) validation, data attestation, and the formation of on-chain transactions. While enclaves themselves are not TEEs, they represent the secure execution context within TEEs, and it is these enclave-resident operations that define the baseline of the Blockchain TCB. This architectural boundary allows us to clearly separate trusted enclave-resident logic from the surrounding untrusted orchestration layer, which includes components like the TC Relay, the SCB, and external data sources. Whenever a secure oracle (either TC or FPC) interacts with the outside world, whether to fetch

data, receive a request, or respond to a query, this interaction inherently traverses an untrusted communication path. For example, in the case of TC, requests are mediated through the TC Relay, a component operating outside the enclave and thus outside the TCB. Similarly, for FPC, even though the chaincode logic is executed within the enclave, it still communicates with off-chain actors via the SCB and Fabric's peer interface, which are not part of the TCB. In what follows throughout this evaluation, all communication flows and operation latencies are measured with a precise understanding of this trust boundary. We explicitly distinguish between timings recorded inside the trusted enclave and those attributed to untrusted orchestration paths, which may introduce variability due to network, OS-level scheduling, or I/O delays. This distinction allows us to better characterize the performance isolation of trusted logic, and to contrast the operational efficiency of TC vs. FPC, especially in how each manages secure communication, policy enforcement, and Blockchain-bound state transitions.

Interoperability and Identity Verification A notable element in REWIRE's methodology is the inclusion of interoperability between secure oracles, an aspect often overlooked in traditional Blockchain benchmarks. REWIRE evaluates the performance impact of data verification and policy enforcement workflows that span multiple SGX enclaves, involving Verifiable Credentials (VCs) and Verifiable Presentations (VPs) in compliance with Self-Sovereign Identity (SSI) principles and W3C. These evaluations reflect how well the system handles decentralized identity proofing and secure delegation within the end-to-end pipeline.

Scalability and Concurrency Testing To complement latency measurements, we also performed scalability assessments under increasing data volumes and parallel access patterns. For read operations, we tested the retrieval of 10, 100, and 1,000 records to evaluate how Blockchain ledger size affects lookup performance. For write operations, which are inherently sequential due to consensus constraints, concurrency was limited to the request processing pipeline and oracle coordination. For concurrent querying, we emulated multiple simultaneous access requests from edge devices to assess the SCB's ability to broker parallel queries without performance degradation. These tests help validate REWIRE's robustness under real-world usage loads in smart city and satellite scenarios.

Confidentiality and Traceability Under Evaluation The methodology also takes into account the confidentiality of datagram transactions, which are a cornerstone of REWIRE's secure data-sharing paradigm. Rather than relying on publicly visible blockchain interactions alone, REWIRE enables authenticated, encrypted off-chain data flows between the SCB and towards SGX enclaves. This setup ensures that sensitive metadata—such as query parameters, device identifiers, and configuration payloads—remain shielded from public exposure while still being verifiable. While TCB size was not explicitly measured in this deliverable, both TC and FPC employ minimal enclave designs, keeping the TCB surface small and auditable.

6.2 Performance Evaluation

Performance evaluation of the REWIRE Blockchain Infrastructure was conducted by simulating operational scenarios aligned with REWIRE's architectural design. Central to this analysis is the **Secure Oracle Layer**.

Evaluation Scenarios Based on Core Architectural Flows Each data management operation was mapped to a representative system scenario derived from REWIRE's architecture (as detailed in Chapter 2). These scenarios align with the previously defined flows and were designed to reflect real-world usage contexts. Each scenario was structured to highlight REWIRE's support for authenticated data feeds, custom datagram management, policy enforcement, and traceability of trust evidence, fulfilling

the requirements for secure and dynamic trust characterization in system-of-systems environments. The scenarios are outlined as follows:

- **Scenario 1 – Application Data Ingestion and Querying**: A swarm of satellites transmits telemetry data periodically to a trusted source. The SCB submits a request for fetching this information to the Besu ledger through the TC enclave. External components or services can query this data through authenticated requests to retrieve the latest or historical application-specific records.
- **Scenario 2 – Attestation Reporting and Retrieval**: Devices periodically undergo attestation report finalization, and results are processed through TC and forwarded for secure anchoring on both Besu and then FPC ledgers. Components retrieve and assess the trustworthiness of a device through authenticated queries.
- **Scenario 3 – SW/FW Update Dissemination**: A centralized control system triggers a one-to-many SW update operation, leveraging Besu to anchor update metadata (e.g., version, hash, release policy) and notify all targeted devices in the swarm. Upon notification, devices initiate follow-up queries to the SCB in order to validate their eligibility based on ABAC-enforced policies and to securely retrieve the corresponding update payload. This process ensures the traceable, verifiable, and policy-controlled distribution of patches across large-scale edge infrastructures, as required in urban deployments or critical network segments. In addition to the broadcast (one-to-many) update mode, REWIRE also supports a one-to-one update modality, where a specific patch is disseminated to a single device or a narrowly scoped group. This approach is applicable in scenarios where individualized updates are required due to device heterogeneity, risk isolation policies, or staged rollout strategies. While functionally similar, the one-to-one modality does not require broad replication or mass notification, and therefore involves lighter transaction and validation footprints. Both modalities share the same trust mechanisms—anchoring, VP verification, and ABAC enforcement, thus supporting consistent policy enforcement and auditability.
- **Scenario 4 – MUD Profile Storage and Enforcement**: MUD profiles are pushed by manufacturers to SCB instances, then stored on-chain via Besu. Upon policy update, SCB notifies associated devices, which subsequently verify and retrieve the new ruleset under ABAC enforcement.

Measurement Objectives and Method The evaluation focused on two principal aspects: (a) the step-by-step **latency per component** during transaction execution, and (b) the overall **end-to-end duration of each operation**. This enabled both granular performance analysis and holistic system-level benchmarking. Step-by-step timing was captured using internal SCB instrumentation and precise timestamp logging during inter-component interactions. Core services, including Besu, TC, and SCB, were co-deployed on the same trusted execution host, ensuring consistent clock baselines and avoiding skew due to distributed measurement errors. This setup enabled us to measure specific execution stages such as: SCB's ABAC policy checks,TC enclave VP signature verification, TC enclave HTTPS data fetches, Besu anchoring latency, and final response delivery. Additionally, we measured the delay for device-triggered queries and FPC interactions for attestation report ingestion and verification.

Scalability and Query Stress Testing Beyond single-operation timings, we examined system behavior under **realistic concurrent workloads** and **large-scale queries**. For example, we evaluated querying performance under different timestamp ranges, essential for retrieving historical attestations, and tested SCB responsiveness to hundreds of simultaneous MUD profile lookups or application data retrievals. This was critical to assess REWIRE's scalability in reputation-driven operations, where the dynamic evaluation of trust over time depends on frequent and timely querying of historical evidence. Importantly, it must be emphasized that queries for application-related data and SW/FW update records do not traverse the TC secure oracle. In REWIRE's design, TC is utilized strictly for its role as a trusted data ingestion oracle, fetching verifiable data from external, off-chain sources and securely anchoring it to the on-chain

ledger. Its usage is tightly scoped to authenticated, verifiable input operations. Querying already-ingested Blockchain data via TC would not only introduce unnecessary overhead but also contradict its core design principle, which is to extend trust boundaries outward and not inward. As such, query operations are efficiently handled by the SCB in direct interaction with the Besu Blockchain, where access control, policy enforcement, and identity verification occur off-chain. This delineation helps maintain a clean separation of responsibilities between data storage and data retrieval, while also enhancing system performance and architectural correctness. The evaluation thus not only captures static performance but also offers insights into how REWIRE behaves under increasing load, mirroring deployment in dense environments such as smart cities or real-time mission networks like smart satellites.

Structure of Evaluation Results In the following subsections, each scenario is evaluated in depth. Flow-specific tables present detailed timing measurements per step, and accompanying workflow diagrams illustrate the orchestration across infrastructure components. These results demonstrate REWIRE's capacity to support secure, policy-enforced data sharing across complex distributed environments, with confidentiality, verifiability, and auditability at their core.

6.2.1 Management of Application-related Data - Scenario 1

Secure Data Ingestion Pipeline The process begins when new data, such as telemetry snapshots, software stack fingerprints, or environmental measurements, is pushed from the application layer to a trusted data source. This source is actively monitored by the SCB, which acts as the orchestrator of all security-sensitive operations. Within a predefined interval, the SCB emits a storage request event to the Hyperledger Besu ledger, triggering the start of the ingestion flow. This Blockchain event is detected by the TC relay, which monitors the Besu ledger for relevant signals. Upon detecting such an event, the relay initiates a secure HTTPS fetch of the associated data using the TC enclave. This enclave operates inside an Intel SGX TEE, ensuring that the data is fetched, parsed, and processed in isolation from potentially compromised environments. Within the enclave, the application data is verified using VPs, including cryptographic signatures and source metadata. Upon successful verification, the data is prepared for Blockchain anchoring. If the payload size is below a predefined threshold (e.g., 20 KB), it is directly stored on-chain in the Besu ledger. For larger payloads, the data is stored in the off-chain storage subsystem (Elasticsearch), and only a hash-based reference is committed to the ledger to ensure integrity and traceability. This entire orchestration, from signal emission to final anchoring, is executed in a policy-aware and auditable manner. It ensures that all participating entities (i.e., SCB, TC, and Besu) operate under verifiable conditions, fulfilling the system's confidentiality, integrity, and access control guarantees.

Policy-Compliant Data Querying In addition to ingestion, REWIRE enables authorized and privacy-preserving querying of application data. Query requests are typically initiated by verified services or higher-level orchestrators. These requests are first intercepted by the SCB, which enforces ABAC policies and verifies the cryptographic signature of the requester. Upon successful validation, the SCB proceeds to retrieve the relevant data. This may involve a direct query to the Besu ledger for on-chain entries or a combined on-chain reference lookup followed by an off-chain fetch for larger entries. The query response is then re-validated, encapsulated, and securely forwarded back to the authorized requester. This policy-enforced, signature-verified, and tamper-resistant querying mechanism supports not only the retrieval of the most recent records, but also historical lookups by timestamp range—enabling traceability, compliance auditing, and situational awareness in real-time systems.

6.2.1.1 Application-related Data Ingestion Benchmark

Evaluation Scope and Step Mapping The end-to-end application data ingestion process, as depicted in Figure 6.1, maps directly to the architectural execution flows introduced in Chapter 2. This workflow

showcases how verifiable application-related data is securely collected, authenticated, and anchored in the REWIRE Blockchain Infrastructure. Each processing stage is benchmarked and corresponds to the measured steps below.

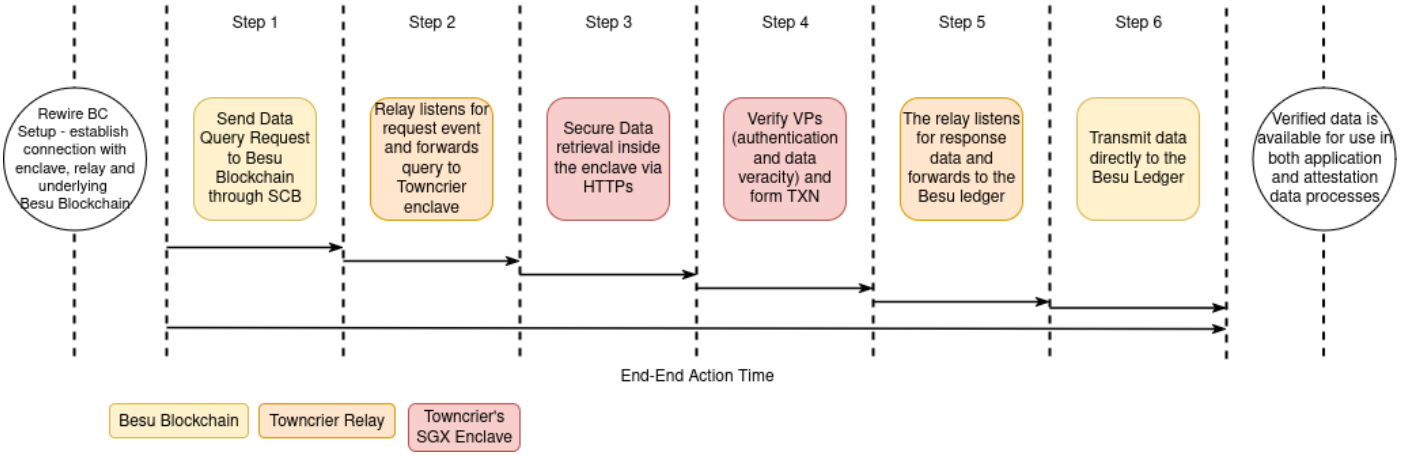

Figure 6.1: Workflow for Application Data Ingestion into the REWIRE Blockchain Infrastructure

- **Step 0: REWIRE BC Setup** - Denotes the preparatory and offline initialization of secure infrastructure connections across core components, including the SGX enclave, the Town Crier relay, and the Besu blockchain node. This foundational setup is not included in execution time measurements.
- **Step 1: Initiate Authenticated Data Feed by Pushing a Data Request Event (via SCB)** - Captures the Initial Query Request via SCB, representing the SCB's emission of a Blockchain event upon detecting the availability of new data at a trusted source. This event kickstarts the off-chain to on-chain transition and serves as the system entry point for authenticated ingestion.
- **Step 2: TC Relay Captures Blockchain Event and Forwards for Processing in TC Enclave** - Measured as Relay Event Forwarding to Town Crier, this asynchronous step bridges the Blockchain and the secure enclave, triggering the secure off-chain data retrieval process.
- **Step 3: Secure Data Retrieval via HTTPS inside TC Enclave** - Aligned with Secure HTTPS Data Retrieval, the enclave fetches the data directly from the trusted source using encrypted HTTPS and processes it within an Intel SGX TEE. This ensures confidentiality, integrity, and resistance to tampering.
- **Step 4: Verification of Presented Attributes as Part of a W3C-Compliant Verifiable Presentation & Transaction Formation** - Captured as VP Verification and Transaction Construction, this stage validates the cryptographic credentials (VPs) of the submitting party, ensuring that the incoming data complies with WSCA-compliant (e.g., ECDSA-based) standards aligned with the EUDI ARF. In this step, device identity and data authenticity are verified before the transaction is formed for Blockchain anchoring.
- **Step 5: Relay Forwards Validated Data to Besu for Storage** - Represented as Response Forwarding, this step handles the transmission of verified and signed data to the Blockchain interface.
- **Step 6: On-Chain Storage to Besu Ledger** - Mapped to On-Chain Storage, this step finalizes the transaction and anchors the data to the Besu ledger. Due to the consensus and block finalization mechanisms of the Blockchain, this step constitutes the majority of the end-to-end latency.

These operations collectively form the authenticated data feed workflow and represent a foundational REWIRE pattern for secure and auditable data transmission. The VP generation in this workflow currently leverages ECDSA-based signatures, ensuring interoperability with emerging W3C Verifiable Credentials

and the EUDI Architecture Reference Framework (ARF). However, it is important to emphasize that this specific instantiation—using only ECDSA—was selected primarily to simplify end-to-end measurement extraction during this evaluation phase. The underlying REWIRE cryptographic agility layer, as detailed in Deliverables D4.2 and D4.3, is designed to support a broader set of standardized cryptographic primitives aligned with the EUDI and SSI ecosystems, including BBS+ signatures, CL signatures, and others. These alternative schemes remain compatible with the REWIRE architecture and can be adopted in future deployments or benchmarking scenarios.

Measured Execution Times The following table summarizes the performance results of each evaluated step. The overall process completes in under 5 seconds, with the on-chain storage dominating the latency, an expected outcome in Blockchain-based systems with strong integrity and auditability guarantees.

Table 6.3: Application Data Ingestion – Performance Evaluation

Step	Execution Time (s)	Definition / Point of Interest
Initial Query invocation via SCB	0.070	**Expected** – This reflects the SCB emitting a Blockchain event upon detecting new data. It involves no backend processing and only event dispatching, so this lightweight timing is fully aligned with expectations.
Relay Event Forwarding to Towncrier	0.022	**Expected** – The TC relay simply forwards the Blockchain event to the SGX enclave. This is basic IPC or messaging overhead with minor marshaling, and the time is well within normal operational bounds.
Secure HTTPS Data Retrieval inside Enclave	0.120	**Expected** – This includes TLS handshake, HTTPS fetching, and enclave-based parsing. Timing is realistic for network-secured data retrieval inside a TEE, and matches performance observed in other secure oracle architectures.
VP Verification and Transaction Construction	0.006	**Expected** – This cryptographic step is accelerated by hardware inside the SGX enclave. The speed reflects efficient signature checking and transaction building, and is noticeably faster than when done outside the TEE (e.g., in SCB).
Response Forwarding for Storage	0.009	**Expected** – Forwarding the verified payload to Besu involves no heavy logic, only marshaling and submission calls, so minimal latency is expected and achieved.
On-Chain Storage to Besu	4.650	**Expected** – This is the dominant latency contributor. PoA consensus introduces natural block production delay. This timing reflects expected block confirmation time in a public-chain-compatible ledger like Besu.
Total End-to-End Duration	4.877	The total duration is consistent with the individual steps. It confirms that secure data consumption is efficient across the trust chain, with on-chain finalization being the bottleneck, not the enclave logic or SCB orchestration.

6.2.1.2 Application-related Data Querying Benchmark

Once application-related data is successfully anchored to the Blockchain, authorized services, such as REWIRE's AI-based misbehavior detection component, can retrieve this data for further processing toward the trust characterization of the target device. This authenticated querying mechanism ensures secure and controlled access to sensitive telemetry and operational records, a critical requirement in

distributed trust architectures. Queries are initiated only by entities that have been verified using W3C-compliant Verifiable Presentations (VPs). The SCB orchestrates this process end-to-end: it authenticates the VP, enforces Attribute-Based Access Control (ABAC) rules, executes the query against the Besu ledger, and securely returns the result. Importantly, REWIRE's design ensures that secure oracles (e.g., Towncrier) are not directly exposed for querying. Instead, the SCB acts as the sole mediator for read operations. This reduces the attack surface while centralizing access enforcement. Although it introduces a single point of mediation, this is balanced by strong authentication and policy enforcement logic within the SCB. Two types of queries are primarily supported: one that fetches the latest entry, and one that retrieves historical data over a timestamp range. These both pass through the same SCB-mediated flow. The end-to-end flow for application queries is illustrated in Figure 6.2.

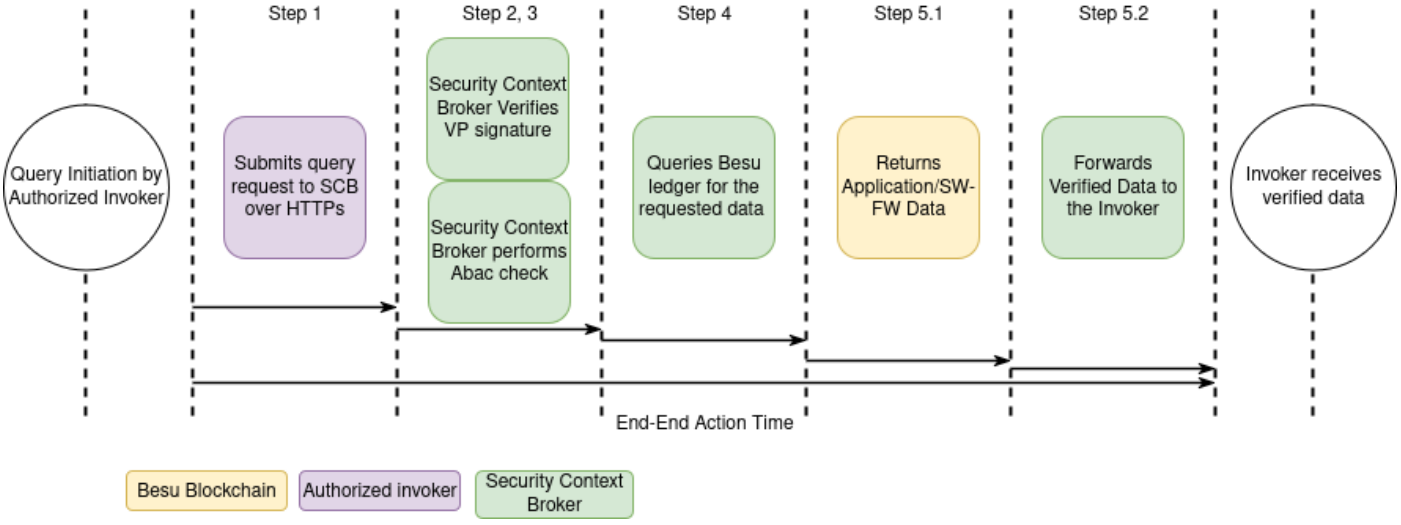

Figure 6.2: Workflow for Application Data Querying in REWIRE

Evaluation Scope and Step Mapping Figure 6.2 represents the complete process for authorized data querying and retrieval within the REWIRE Blockchain infrastructure. This diagram maps directly to flows of the architecture defined in chapter 2, which governs secure data access by authenticated entities. The steps correspond precisely with the evaluation metrics for querying application-related or SW/FW-related data.

- **Step 0: Query Initiation by Authorized Invoker** → The querying process begins with an authorized invoker (application or device) initiating a request to the Security Context Broker (SCB). This is a logical entry point and not individually timed but essential to delineate the transaction's start.
- **Step 1: Submit Query Request to SCB over HTTPS** → Corresponds to the secure submission of the query to the SCB. The execution latency of this submission is included in the Initial Query Invocation via SCB.
- **Step 2: SCB Performs VP Signature Verification** → Aligned with VP Signature Verification (SCB) (Step 2 in the table), this step authenticates the requester by verifying cryptographically signed Verifiable Presentations(ECDSA).
- **Step 3: SCB Executes ABAC Enforcement** → This step is represented by ABAC Enforcement (SCB) (Step 3), where the system checks policy compliance based on the invoker's attributes.
- **Step 4: SCB Queries Besu Ledger** → Mapped to Blockchain Query Execution (Besu) or Timestamp-Based Lookup (Besu) (Step 4), this covers the Blockchain-level search.

- **Step 5.1: On-Chain Data Returned to SCB** → This step is internal to Step 5 and reflects Besu's response back to the SCB. It is not independently measured but is included in the recorded Blockchain query execution times.
- **Step 5.2: SCB Forwards Verified Data to Invoker** → Corresponds to Response Forwarding to Invoker (Step 5), denoting the secure return of filtered, policy-validated data to the requester.

Measured Execution Times Measured timings for these operations reflect the efficiency of the querying mechanism. For queries that retrieve only the latest entry see Table 6.4.

Table 6.4: Query Type: Get Latest Application Data

Step	Execution Time (s)	Definition / Point of Interest
Initial Query invocation via SCB	0.070	**Expected** – This is consistent with typical HTTPS round-trip time for authenticated requests with moderate payloads. There is no backend computation at this point.
VP Signature Verification (SCB)	0.042	**Slightly Slower than TEE-based** – While this time is reasonable, it is slightly higher than what we observe inside SGX enclaves due to lack of hardware acceleration.
ABAC Enforcement (SCB)	0.050	**Expected** – This involve parsing the attributes and matching them against policy rules. It is a lightweight logic step but involves some computation and is within expected bounds.
Blockchain Query Execution (Besu)	0.060	**Expected** – Read operations on Besu do not involve consensus. The timing aligns with querying indexed key-value pairs on-chain and fetching state data.
Response Forwarding to Invoker	0.020	**Expected** – Simple HTTPS response forwarding with negligible transformation or logic involved. Latency is network-bound and well within normal parameters.
Total End-to-End Duration	0.242	The full process staying under 250 ms validates REWIRE's design for near-real-time responses in edge computing.

For broader historical lookups by timestamp range see Table 6.5.

Table 6.5: Query Type: Timestamp Range Retrieval

Step	Execution Time (s)	Definition / Point of Interest
Initial Query invocation via SCB	0.073	**Expected** – This includes the HTTPS submission of a query request by the client. The timing reflects network and parsing overhead and is in line with what is expected for typical secure client-server interaction.
VP Signature Verification (SCB)	0.041	**Slightly Slower than TEE-based** – While this time is reasonable, it is slightly higher than what we observe inside SGX enclaves due to lack of hardware acceleration.
ABAC Enforcement (SCB)	0.048	**Expected** – This step involves policy rule matching and identity attribute parsing. The duration is reasonable and reflects expected access control overhead within a policy broker operating in user space.

Table 6.5: Query Type: Timestamp Range Retrieval

Step	Execution Time (s)	Definition / Point of Interest
Timestamp-Based Lookup in Besu	0.109	**Expected** – Searching time-indexed entries in Besu requires traversal of the ledger's state. The slightly higher duration here is logical given the range-based search and potential size of the on-chain dataset.
Response Forwarding	0.016	**Expected** – This is a simple return of the filtered results to the client over HTTPS. Minimal latency is anticipated and observed, consistent with previous forwarding operations.
Total End-to-End Duration	0.287	The total duration reflects the cumulative latency of access control, query execution, and response handling. The result confirms that querying via SCB and Besu remains sub-300ms under realistic workloads.

To further evaluate scalability, various query payloads were tested see Table 6.6.

Table 6.6: Scalability of Application Data Querying

Number of Returned Entries	Execution Time (s)
10	0.109
100	0.134
500	0.172
1000	0.238
2000	0.384
5000	0.646
10000	1.211
20000	2.472

These results validate that the REWIRE Blockchain infrastructure exhibits strong scalability properties when handling large volumes of data retrieval operations. Even under high query loads, performance degradation remains linear and manageable, without sudden spikes or instability. This outcome highlights the efficient integration between SCB-mediated access control and Besu ledger indexing, allowing high-throughput querying while preserving fine-grained access control. Moreover, it demonstrates the system's ability to support authenticated and identity-aware data access even at scale. Beyond raw performance, these measurements underscore a key advantage of the REWIRE dual-pipeline architecture:

- **Ingestion operations**, which are anchored and verified through secure enclaves (Towncrier SGX), ensure strong provenance, authenticity, and tamper-resistance of all ingested data.
- **Query operations**, which are mediated by the SCB and executed on Besu, maintain responsiveness by bypassing the enclave path, thereby optimizing for read efficiency without sacrificing security.

This architecture achieves a balanced interplay between security assurances and performance guarantees. By decoupling ingestion and querying responsibilities across trusted and efficient components respectively, REWIRE avoids the common bottlenecks found in enclave-heavy pipelines while still maintaining verifiability.

6.2.1.3 Observations and Insights

The evaluation of the authenticated data feed workflow highlights the practical feasibility of using secure, distributed ledger infrastructures to mediate high-assurance data exchange without compromising the responsiveness or safety profile of the target applications. This subsection documents the key findings and underlying architectural choices that make such a balance possible.

Latency Origins and Security Trade-offs The measured latency in the application data ingestion workflow confirms that the **primary contributor to overall delay is the on-chain storage** step, with Hyperledger Besu typically requiring ~ 4.65 seconds for transaction finalization. This latency originates from the block confirmation interval enforced by the Proof-of-Authority (PoA) consensus mechanism, as further discussed in the performance breakdown of consensus protocols. By contrast, the **orchestration overhead** introduced by the SCB, the TC Relay, and SGX enclave operations, including secure HTTPS retrieval and VP validation, remains minimal, amounting to **less than one second** in total. This highlights an important trade-off: while on-chain anchoring ensures transparency and immutability, the cryptographic assurances (e.g., VP verification and enclave-protected execution) can be achieved with low overhead and without adversely impacting responsiveness. These results demonstrate that REWIRE's integration of trusted computing and privacy-preserving mechanisms into the data ingestion pipeline preserves system efficiency while providing robust guarantees of confidentiality, integrity, and verifiability. Application-related data queries, which are mediated exclusively by the SCB and executed over Besu, consistently demonstrate sub-300ms end-to-end execution times. This holds true for both latest-entry retrievals and timestamp-range queries. The efficiency of the SCB's policy enforcement path, including VP signature checks and ABAC rule evaluation, ensures that secure access control does not impose substantial delays. This responsiveness enables real-time decision-making by downstream components, such as REWIRE's AIMDE engine.

Scalability and Predictable Behavior Experiments across **varying query sizes** (10 to 20,000 entries) indicate a linear relationship between result size and response time. There are no sudden latency spikes or anomalies under increased load, validating the system's scalability under realistic access patterns. This behavior is particularly important for long-running deployments where historical data must be queried to assess the evolution of a device's trustworthiness. The system's ability to respond predictably and efficiently to such requests is key to supporting dynamic trust characterization models in REWIRE.

Architectural Separation and its Benefits A deliberate design decision in REWIRE is the separation of ingestion and querying concerns. Application data is verified and ingested through Towncrier's secure SGX enclave, ensuring data authenticity, while querying is handled directly via the SCB and Besu, prioritizing speed. This division of trust enforcement at write-time and efficiency at read-time enables REWIRE to preserve strong verifiability guarantees without burdening retrieval operations. It also provides a clean trust interface between privacy-sensitive ingestion paths and performance-sensitive consumer logic.

Architectural Separation and Its Benefits A key architectural principle in REWIRE is the deliberate **separation between data consumption and querying workflows**. Application-related data is verified and securely ingested through the TC's SGX enclave, ensuring authenticity and cryptographic integrity at the moment of submission. In contrast, querying operations are executed directly via the SCB and the Besu ledger, bypassing the TC and prioritizing minimal latency. This separation is not arbitrary. It is rooted in a practical trade-off between trust enforcement at write-time and performance-critical access at read-time. Since data authenticity has already been cryptographically verified and anchored on-chain during consumption, querying operations can safely access this evidence without repeating enclave-based validation. This architectural decoupling ensures that verifiability is preserved, while maximizing responsiveness for downstream consumers. The choice to avoid routing queries through the TC is important in safety-critical contexts, such as automotive collision avoidance or smart infrastructure control, where real-time trust-based decisions must be made in under 200 milliseconds, as mandated by domain-specific latency thresholds. In such scenarios, access to the latest verified evidence must incur negligible overhead; otherwise, the trust assessment process itself becomes infeasible for real-time response. By enabling direct and policy-compliant querying through the SCB, REWIRE achieves near-instantaneous access to trust-related information. This approach allows the system to support dynamic trust evaluation while remaining within the bounds of stringent timing requirements. The current architecture already

approaches these thresholds, and future optimization of query handling and indexing layers can further improve performance for high-frequency trust-driven applications.

Support for Safety-Critical Automation By maintaining low-latency query paths and secure write flows, the REWIRE Blockchain infrastructure supports timely actuation and detection workflows, which are critical in cyber-physical environments. Application-layer logic, such as anomaly detection or automated policy adjustments, can confidently act on authenticated and trustworthy data, without incurring delays that could compromise system safety or responsiveness.

6.2.2 Storing and Indexing Trust-related Evidence - Scenario 2

A core aspect of the REWIRE architecture is the trusted recording and retrieval of attestation-related evidence. These records serve as cryptographically verifiable claims about the software integrity, operational posture, and policy compliance of edge devices. The capability to store and index such evidence is foundational to enabling dynamic trust characterization, auditability, and automated enforcement decisions across the lifecycle of connected devices. The REWIRE infrastructure supports this functionality through the coordinated operation of two secure oracle systems, TC and FPC, alongside the SCB and Besu as the underlying ledger. In general, REWIRE aims to allow direct interaction between TC and FPC, forming a trust-preserving pipeline in which TC collects or ratifies verifiable information and passes it securely to FPC for final processing and policy-bound storage.

Architectural Intent vs. Implementation Constraints In the conceptual design of REWIRE, the TC and FPC oracles were envisioned as directly interoperable SGX-enabled components, exchanging verifiable attestation data through a confidential and authenticated enclave-to-enclave channel. This direct pipeline would reduce latency, minimize the attack surface, and strengthen the verifiability of the trust chain. However, due to technical constraints in the current implementation, this direct oracle-to-oracle handover was not fully realized. TC relies on a legacy SGX SDK with asynchronous fetch logic, while FPC operates on Hyperledger Fabric's chaincode model with a distinct transaction and endorsement architecture. These differences made direct integration impractical in this phase. As a result, the prototype follows an intermediate integration path: attestation evidence is validated inside TC, anchored on the Besu Blockchain, and later retrieved by the SCB for final verification and policy-enforced storage in the FPC ledger. While this adds a step, it still maintains SGX-based confidentiality and end-to-end integrity. The performance measurements in this section thus reflect this practical pipeline, from initial SGX-based validation through on-chain anchoring and final persistence, offering realistic insight into REWIRE's current operational guarantees and future extensibility.

Workflow Overview The process begins when a target edge device undergoes a remote attestation check. The outcome, typically in the form of a W3C-compliant VP signed by the device's secure element, and a notification is pushed to the SCB. The SCB triggers a data ingestion signal to TC, which fetches the data from the device, validates the VP inside its enclave and constructs a transaction for storage on the Besu ledger. Once this anchoring step completes, the SCB signals the FPC enclave to query Besu for the newly available attestation evidence. The FPC component performs a second layer of validation, persists the evidence inside its trusted chaincode storage, and applies ABAC policies to determine the contextual integrity of the device. This two-phase approach ensures that attestation records are both anchored immutably on-chain and securely re-evaluated inside an SGX-enforced runtime, achieving end-to-end verifiability despite the temporary architectural gap between the two oracle systems. The measured execution times and the full step mapping of this pipeline are detailed in the next subsection.

6.2.2.1 Attestation Report Ingestion Benchmark

The end-to-end flow for storing attestation reports begins with the TC enclave fetching attestation data from trusted edge devices. After verifying the integrity and origin of the data through its VPs, the enclave sends the data to the relay for either on-chain storage (if the payload is small) or off-chain storage with a pointer to Besu. Once the data is successfully stored on Besu, the SCB emits a notification to the FPC ledger via WebSocket. The FPC, acting as a Besu client, then queries the stored object from the Blockchain, verifies the VP again using its internal signature verification logic, and finally stores the validated attestation evidence inside its ledger. The process is illustrated in Figure 6.3.

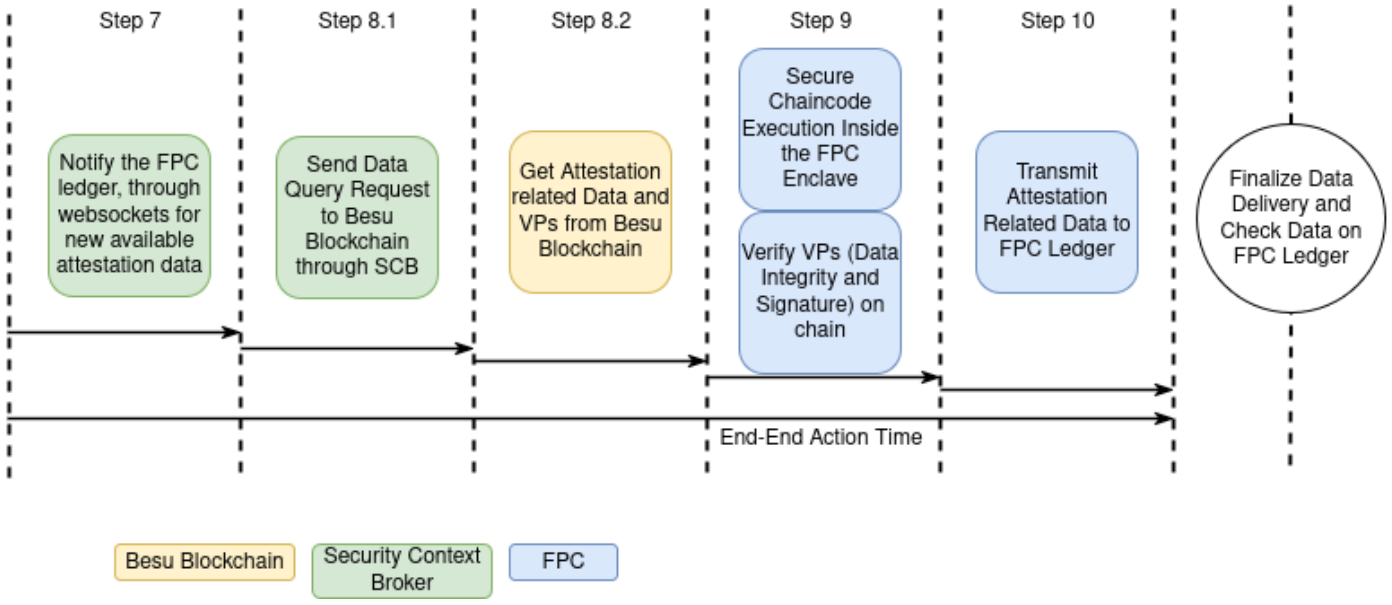

Figure 6.3: Secure Storage of Attestation Data into FPC Ledger

Evaluation Scope and Step Mapping The attestation data ingestion process, illustrated in Figure 6.3, is aligned with the architectural flows defined in Chapter 2 and the metrics captured in the corresponding performance evaluation table. Each action in the diagram is mapped to a concrete evaluation step as follows:

- **Step 1: Send Data Query Request to Besu Blockchain through SCB** → This corresponds to Initial Data Query Request (Step 1) in the table. It includes secure HTTPS submission from the SCB and preliminary data preparation operations.
- **Step 2: Relay listens for request event and forwards query to Town Crier enclave** → Captures the Relay Forwarding operation (Step 2), representing asynchronous message dispatching from SCB to the enclave via the relay service.
- **Step 3: Secure Data Retrieval inside the enclave via HTTPS** → Represents Step 3, in which external attestation data is retrieved securely inside the SGX-protected enclave.
- **Step 4: Verify VPs (authentication and data veracity) and form TXN** → Mapped to VP Verification & Transaction Formation (Step 4), covering cryptographic checks and transaction assembly before anchoring.
- **Step 5: Response Forwarding from enclave back to relay** → Denotes the delivery of verified data to be relayed toward the Blockchain backend.
- **Step 6: Transmit data directly to the Besu Ledger** → Corresponds to On-Chain Storage (Besu) (Step 6), reflecting the latency of Blockchain write operations including smart contract execution.

- **Step 7: Notification via WebSocket to FPC Ledger** → Aligns with Step 7, where the SCB emits a WebSocket message notifying the FPC layer of new attestation data availability.
- **Step 8: Query and Retrieve from Besu** → The FPC subsystem issues a request to fetch the new data. This corresponds to Step 8, which involves Blockchain read access.
- **Step 9: Signature Verification in FPC Enclave** → This step ensures that the data fetched from Besu maintains its integrity and authenticity before inclusion in the FPC ledger.
- **Step 10: Final Storage on FPC Ledger** → Captures the final stage where attestation data is securely persisted inside the FPC ledger via enclave execution.

These steps capture the full cross-ledger attestation data lifecycle, integrating Town Crier, Besu, and FPC subsystems under the REWIRE architecture. Each measurement reflects the latency implications of secure, integrity-verified storage and cross-chain propagation of trust-related evidence.

Measured Execution Times The full breakdown of execution times for each of these actions is shown in Table 6.7.

Table 6.7: Attestation Data Ingestion – Performance Evaluation

Step	Execution Time (s)	Definition / Point of Interest
Initial Data Query Request	0.070	**Expected** — This reflects the SCB emitting an authenticated event to initiate the process. No back-end execution or heavy computation is involved, so the latency is minimal and in line with event emission overhead.
Relay Forwarding	0.026	**Expected** — A lightweight inter-process relay handoff. The latency is consistent with simple marshaling and message passing from the relay to the enclave.
Secure Data Retrieval (HTTPS)	0.132	**Expected** — This includes TLS handshake and secure data retrieval within the enclave. The time aligns with typical HTTPS fetch + SGX enclave overhead and data unmarshalling.Expected — This includes TLS handshake and secure data retrieval within the enclave. The time aligns with typical HTTPS fetch + SGX enclave overhead and data unmarshalling.
VP Verification & Transaction Formation	0.008	**Expected** — This is fast due to hardware-accelerated ECDSA verification inside the SGX enclave. The cryptographic workload is small, and enclave-based execution is efficient, especially compared to SCB-based verification later.
Response Forwarding	0.011	**Expected** — A non-computational step involving simple message handoff from enclave back to SCB. The latency is minimal as anticipated.
On-Chain Storage (Besu)	4.71	**Expected** — This is the dominant contributor, as discussed in the consensus analysis subchapter. The latency reflects block production under Proof of Authority (PoA), with a typical block interval of 5 seconds.
Notification via WebSocket	(Included above)	Executed asynchronously; not directly contributing to end-to-end latency. Its exclusion from direct timing is valid and expected.

Table 6.7: Attestation Data Ingestion – Performance Evaluation

Step	Execution Time (s)	Definition / Point of Interest
Query and Retrieve from Besu	0.091	**Expected** — Includes smart contract log query + RPC interface marshaling. The timing matches similar query flows and is fast enough for near real-time trust evaluation.
Signature Verification in FPC Enclave	0.083	**Expected** — While slightly slower than TC enclave verification, this reflects FPC's use of Go-based chaincode logic and the more complex policy-checking structure. Still, it remains within acceptable bounds.
Final Storage on FPC Ledger	0.086	**Expected** — Reflects internal secure write operations into the private ledger of FPC. This timing aligns with observed FPC consensus and enclave processing latency under moderate load.
Total End-to-End Time	5.217	The overall timing reflects a logical accumulation of all steps, dominated by on-chain anchoring. It confirms REWIRE's ability to process full trust cycles within a bounded and verifiable timeline.

The notification step (Step 7) acts as a critical bridge that connects both paths (write & read). While it is logically distinct, it does not introduce measurable latency on its own. Instead, it serves to ensure timely awareness on the device side without forcing constant polling or unnecessary interaction with the Blockchain layer. This latency remains consistent under various conditions and is driven mostly by the latency of on-chain consensus in Besu and the cryptographic verification steps within both TC and FPC enclaves.

6.2.2.2 Trust-related Evidence Querying from FPC Benchmark

Once attestation reports are stored, authorized components may query them for risk evaluation, trust scoring, or analytics. The SCB acts as the entry point for all queries, enforcing both signature verification and attribute-based access control (ABAC) before forwarding the query to FPC. The queries may target the latest attestation report for a device or request the complete set of attestation logs associated with a device ID.

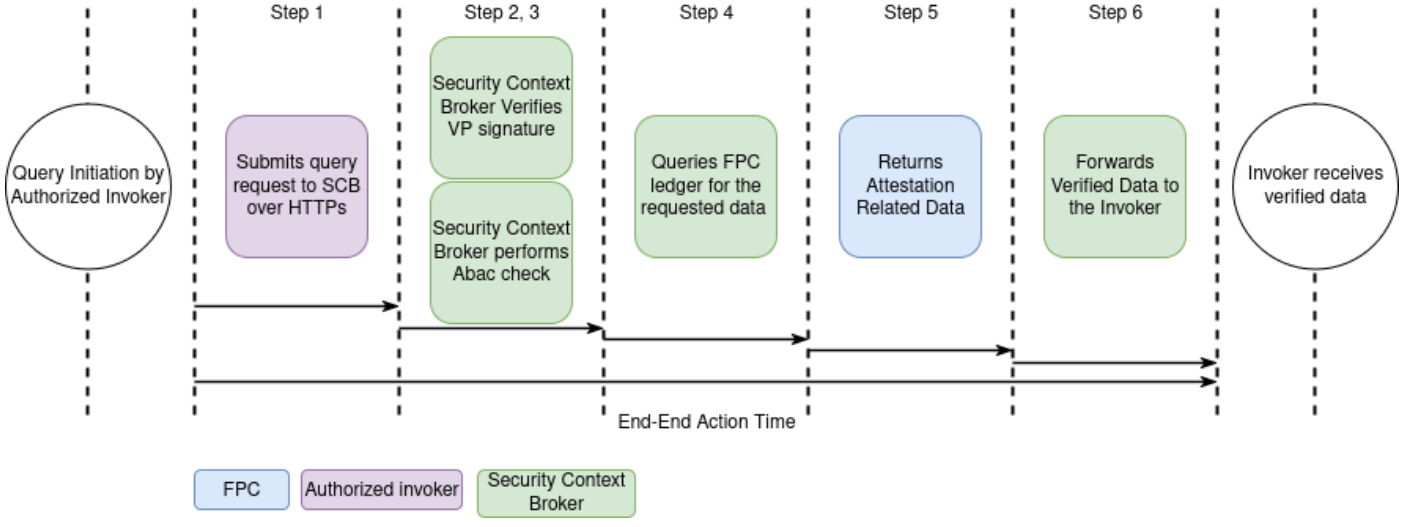

Figure 6.4: Attestation Query Workflow via Security Context Broker and FPC

Evaluation Scope and Step Mapping The attestation data retrieval workflow illustrated in Figure 6.4 directly reflects the secure and policy-governed data access mechanisms defined in REWIRE's architecture (see Chapter 2.1). Each logical operation in the sequence diagram corresponds to a measurable evaluation step, as described below:

- **Step 0: Query Initiation by Authorized Invoker** → Denotes the conceptual starting point of the query process, where a permitted actor initiates the attestation retrieval. This step is not timed independently but defines the boundary of the evaluation.
- **Step 1: Submits query request to SCB over HTTPS** → This maps to the Initial Query Invocation via SCB step in the evaluation tables. It includes the transmission overhead and initial parsing by the SCB.
- **Step 2: Security Context Broker Verifies VP signature** → Refers to Signature Verification (SCB), ensuring that the requester is cryptographically validated.
- **Step 3: Security Context Broker performs ABAC check** → Captures ABAC Enforcement, where the SCB enforces attribute-based access control before granting data access.
- **Step 4: Queries FPC ledger for the requested data** → Mapped to Query to FPC Enclave (Get Latest or Get All), where a secure read operation is initiated inside the FPC enclave.
- **Step 5: Returns Attestation Related Data** → This event is functionally covered by the prior step and overlaps with ledger response handling within the enclave.
- **Step 6: Forwards Verified Data to the Invoker** → Aligned with Response Forwarding to Requester, this is the final message transmission from the SCB back to the querying entity.

Measured Execution Times The performance metrics for each are presented in Tables 6.8 and 6.9 below, where two key query types were evaluated: (a) Get Latest Attestation Report for Device ID and (b) Get All Reports for Device ID.

Table 6.8: Query Type: Get Latest Attestation Report for Device ID

Step	Execution Time (s)	Definition / Point of Interest
Initial Query Invocation via SCB	0.070	**Expected** — This step includes the HTTPS submission of the query by an authenticated component. The latency is consistent with standard secure API invocation over HTTPS and represents no unexpected delays.
Signature Verification (SCB)	0.040	**Slightly Slower than TEE-based** – While this time is reasonable, it is slightly higher than what we observe inside SGX enclaves due to lack of hardware acceleration.
ABAC Enforcement (SCB)	0.045	**Expected** — Access policy evaluation using attribute-based filters over parsed VP content. This involves basic logical operations and policy matching; the time is appropriate for JavaScript-based processing.
Query to FPC Enclave (Get Latest)	0.088	**Expected** — This latency reflects a secure internal query inside the FPC enclave. It aligns with previously recorded read times and includes enclave entry/exit overhead and secure data unmarshalling.

Table 6.8: Query Type: Get Latest Attestation Report for Device ID

Step	Execution Time (s)	Definition / Point of Interest
Response Forwarding to Requester	0.018	**Expected** — Lightweight network forwarding of a signed response to the requestor. The time is minimal and typical for short, unidirectional data return over HTTPS with no processing attached.
Total End-to-End Time	0.261	The overall latency for retrieving the latest attestation record remains well under the 200 ms threshold for safety-critical operations and matches the performance profile established for SCB-FPC workflows.

Table 6.9: Query Type: Get All Attestation Reports for Device ID

Step	Execution Time (s)	Definition / Point of Interest
Initial Query Invocation via SCB	0.073	**Expected** — This includes the initial HTTPS request to fetch attestation history. The latency is consistent with secure API request submission, involving no back-end or computational overhead at this stage.
Signature Verification (SCB)	0.041	**Slightly Slower than TEE-based** – While this time is reasonable, it is slightly higher than what we observe inside SGX enclaves due to lack of hardware acceleration.
ABAC Enforcement (SCB)	0.046	**Expected** — The latency reflects policy evaluation based on attribute filtering. Processing time is logical, given real-time evaluation of structured identity data and rule matching logic within SCB.
Query to FPC Enclave (Get All)	0.090	**Expected** — Querying a full device attestation history from within the enclave shows consistent and acceptable latency, especially considering secure memory access and enclave boundary transitions.
Response Forwarding to Requester	0.019	**Expected** — Lightweight HTTPS response delivery from SCB to requester. No post-processing involved, and time is consistent with basic network transmission.
Total End-to-End Time	0.269	This end-to-end latency is within acceptable thresholds for bulk trust-related data access and confirms the system's suitability for dynamic trust evaluation workflows, even under full history retrieval.

The results demonstrate a consistently low-latency response time, even when querying the full set of attestation reports for a specific device. This highlights the scalability of the FPC ledger and the efficiency of SCB as an orchestration layer.

6.2.2.3 Observations and Insights

The results of this evaluation phase validate the robustness, efficiency, and scalability of REWIRE's trust-related evidence management infrastructure. Specifically, the end-to-end storage operation, from the initial trigger through SCB, to cryptographic validation, and final anchoring in both Besu and FPC, demonstrates that strong guarantees of verifiability and confidentiality can be achieved with manageable overhead. The full ingestion workflow completes in approximately 5.217 seconds, the bulk of which (~ 4.71 seconds) is attributed to the Besu ledger's transaction finalization latency. This is consistent with expectations in Ethereum-based networks, where consensus-driven confirmation times introduce predictable

delays. Importantly, the remaining steps, which include inter-component notification via WebSocket, secondary retrieval and validation of attestation evidence, signature verification within the FPC enclave, and final secure persistence into the FPC ledger, add less than 1.5 seconds of total latency. This underscores the optimization and careful orchestration of the cross-oracle trust pipeline, which involves both TC and FPC. It is also crucial to highlight that this evaluated architecture represents a first-phase deployment, where interaction between the two secure oracles is still routed through Besu due to technology constraints. Direct cross-oracle communication, as envisioned in the REWIRE conceptual architecture, remains a future goal and will require further abstraction or inter-enclave bridging mechanisms. The current design, though indirect, still allows for layered verifiability and progressive trust anchoring.

On the querying side, attestation lookups, whether for the latest report or for the complete device history, are executed exclusively within the SGX-backed FPC ledger. These operations consistently remain under 250 milliseconds, even when fetching large sets of logs. This performance confirms the viability of using FPC as a scalable, low-latency attestation repository, capable of supporting real-time trust evaluation pipelines, such as dynamic device scoring or compliance checks. Notably, REWIRE's design ensures that these trust queries, especially those supporting mission-critical or safety-sensitive actions, remain as close as possible to the threshold defined by real-time standards (e.g., under 200 milliseconds [1]) for automated decision-making systems. This includes scenarios such as collision avoidance or autonomous reconfiguration, where the rapid assessment of device trustworthiness is vital. The current results indicate that REWIRE is near this performance target, and with further optimization, fully real-time, policy-based trust decisions can be achieved without compromising data verifiability or confidentiality. The SCB plays a pivotal role in maintaining data access integrity. By enforcing both W3C-compliant VP checks and ABAC policies. The SCB ensures that only authenticated and authorized actors can retrieve sensitive attestation data. Notably, this access control overhead is minimal, adding roughly 0.09 seconds to total query time, showcasing that fine-grained identity- and attribute-aware authorization can be achieved without performance compromise. It is important to note that VP verification is executed in two distinct enclave environments, each with its own runtime and design constraints:

- In **TC**, the VP is validated inside a **C-based SGX enclave**, which performs lightweight signature verification logic within a synchronous data fetch pipeline. The TC enclave's design is optimized for compact, stateless validation of incoming data fetched from off-chain sources.
- In **FPC**, VP verification takes place inside a **Go-based chaincode execution context**, integrated within the Fabric Trusted Chaincode (TCC) framework. This environment enforces endorsement policies and transaction logic within a broader ledger consistency model. As such, VP handling in FPC is more tightly coupled with Fabric's state model and cryptographic endorsement mechanism, making it heavier and more policy-bound but also more auditable and deterministic.

This difference is not merely implementation detail, but it reflects distinct security boundaries and execution semantics. While TC's enclave excels at low-latency, real-time proof checking of dynamic data, FPC's enclave supports persistent ledger commitments with endorsement and compliance guarantees. Together, these roles complement one another and provide REWIRE with both flexibility and depth in trust validation. Finally, REWIRE's dual-verification mechanism, where cryptographic evidence (VPs) is first verified in the TC enclave during ingestion and then re-verified within the FPC enclave during querying, embodies the project's commitment to defense-in-depth. This layered approach ensures that no single verification step is implicitly trusted, aligning with Zero Trust principles and elevating the overall integrity of the trust data pipeline.

6.2.3 Threat Intelligence Data Management - Scenario 3

The secure management and distribution of SW/FW updates play a critical role in maintaining trust and resilience across REWIRE's edge computing infrastructure. This subchapter evaluates the end-to-end in-

gestion, verification, and delivery of SW/FW patches, which are treated as threat intelligence artifacts enriched with verifiable integrity proofs and strict access control policies. The focus of the current evaluation lies in the one-to-many dissemination modality, in which a centralized control system initiates a SW/FW update intended for a group of authorized edge devices. This form of distribution is particularly relevant in decentralized smart city deployments and IoT swarms, where parallel patching is needed across heterogeneous environments. A one-to-one variant, targeting individual devices (e.g., for emergency fixes or targeted security enforcement), follows the same processing pipeline and security guarantees, differing only in the scope of its ABAC filtering and delivery endpoint.

6.2.3.1 SW/FW Update Flow Benchmark

The secure update flow involves both off-chain and on-chain mechanisms: SGX enclaves validate update integrity, the SCB enforces attribute-based policies, and the Blockchain ensures traceability and auditability of the delivered patches. Figure 6.5 provides an overview of this secure and verifiable distribution process, illustrating the integration of cryptographic assurance, policy enforcement, and decentralized coordination that underpins REWIRE's update pipeline.

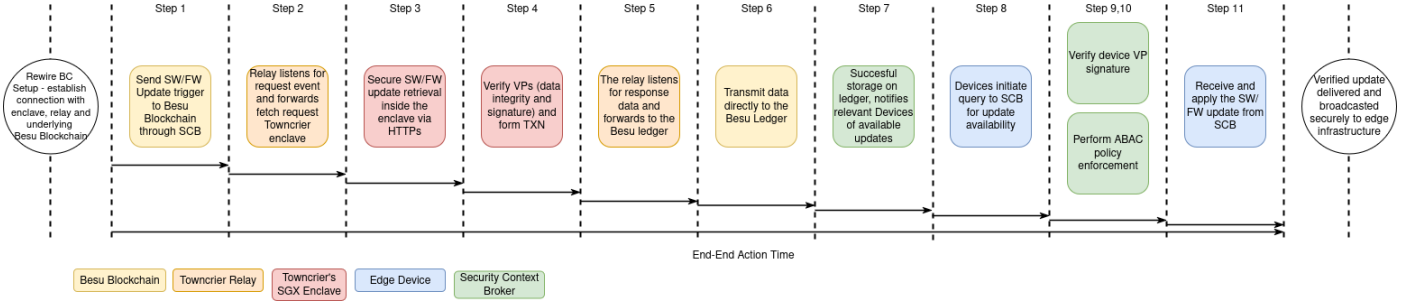

Figure 6.5: Secure SW/FW Update Flow in REWIRE

Evaluation Scope and Step Mapping The SW/FW update ingestion flow presented in Figure 6.5 mirrors the architectural mechanisms defined in Chapter 2 and illustrates the combined write-and-notify process for verified SW/FW update in REWIRE. The evaluated scenario reflects the one-to-many distribution modality, while also being representative of the one-to-one case due to the identical processing pipeline. Each step in the figure maps to a specific evaluation point as follows:

- **Step 0: REWIRE BC Setup** → Denotes the preparatory and offline initialization of secure infrastructure connections across core components, including the SGX enclave, the Town Crier relay, and the Besu blockchain node. This foundational setup is not included in execution time measurements.
- **Step 1: Send SW/FW Update Trigger to Besu Blockchain through SCB** → Initial Data Query Request in the evaluation table. This step marks the triggering of the patch distribution flow by the SCB.
- **Step 2: Relay listens for request events and forwards fetch request to Town Crier enclave** → Relay Forwarding, handling asynchronous propagation of the event for patch acquisition.
- **Step 3: Secure SW/FW update retrieval inside the enclave via HTTPS** → Secure Data Retrieval (HTTPS), capturing secure communication inside the Town Crier SGX enclave for fetching the update artifact.
- **Step 4: Verify VPs (data integrity and signature) and form TXN** → VP Verification & Transaction Formation, reflecting ECDSA signature validation and the construction of the Blockchain transaction. The VP follows W3C-compliant credential structures and is part of REWIRE's SSI-based identity framework.

- **Step 5: The relay listens for response data and forwards it to the Besu ledger** → Response Forwarding, which dispatches validated content for anchoring on-chain.
- **Step 6: Transmit data directly to the Besu Ledger** → On-Chain Storage (Besu), representing the Blockchain-level finalization of the SW/FW patch update using PoA consensus.
- **Step 7: Successful storage on ledger, notifies relevant devices of available updates** → Notification via SCB to Edge Devices. This event uses a WebSocket channel for device notification and is not directly timed.
- **Step 8: Devices initiate query to SCB for update availability** → Not individually measured but forms the beginning of the ABAC-controlled update request workflow within the SCB.
- **Step 9: Verify device VP signature** → Signature Verification of Device (SCB), where the querying device's authenticity is validated.
- **Step 10: Perform ABAC policy enforcement** → ABAC Enforcement (SCB), where access policies are checked to ensure the device is eligible to receive the patch.
- **Step 11: Receive and apply the SW/FW update from SCB** → Patch Query & Delivery to Device, completing the secure and verified update transaction to the endpoint.

Measured Execution Times The complete lifecycle of the SW/FW update process was benchmarked under realistic conditions reflecting REWIRE's use case scenarios. The measured workflow includes both the ingestion of verified patch artifacts and their secure delivery to authorized edge devices. The timing breakdown presented in Table 6.10 corresponds directly to the evaluated step mappings described earlier in this subchapter. It is important to highlight that the notification step (Step 7), does not introduce measurable latency. Instead, it serves as an asynchronous signalling mechanism allowing edge devices to promptly detect new patch availability, avoiding inefficient polling and reducing Blockchain interaction frequency. This design choice strengthens both scalability and reactivity in REWIRE's trusted update dissemination framework.

Table 6.10: SW/FW Update Data Ingestion

Step	Execution Time (s)	Definition / Point of Interest
Initial Data Query Request	0.070	**Expected** — This marks the SCB's submission of the initial update trigger to the Besu ledger. The latency is consistent with a basic HTTPS submission and minimal pre-processing.
Relay Forwarding	0.027	**Expected** — The TC relay asynchronously forwards the Blockchain event to the enclave. This is a lightweight step with low overhead and no cryptographic or access control involvement.
Secure Data Retrieval (HTTPS)	0.130	**Expected** — Secure HTTPS fetch from a trusted source within the enclave. The latency includes TLS setup and internal parsing. It falls well within anticipated bounds for SGX-protected external data retrieval.
VP Verification & Transaction Formation	0.008	**Faster than SCB-side** — This step is executed inside an SGX enclave with hardware crypto acceleration. Time is minimal due to efficient handling of ECDSA verification and transaction construction.
Response Forwarding	0.010	**Expected** — Represents a simple forwarding operation from the enclave to Besu. No computation involved; latency is minimal and predictable.

Table 6.10: SW/FW Update Data Ingestion

Step	Execution Time (s)	Definition / Point of Interest
On-Chain Storage (Besu)	4.715	**Expected** — Dominant latency source due to PoA consensus and block finalization delay. Falls in line with known Besu write performance, as discussed in Section 6.3.3.
Notification via SCB to Edge Devices	-	This WebSocket notification step is asynchronous and not critical-path. It's not measured separately due to negligible and variable latency outside SCB control.
Device initiates Query to SCB	0.070	**Expected** — Represents the HTTPS request initiation by the device once it detects the notification. Comparable to other query initiations; no significant variation.
Signature Verification of Device (SCB)	0.071	**Slightly Slower than TEE-based** — VP verification at the SCB is done without hardware acceleration, making this step marginally slower than enclave-based counterparts.
ABAC Enforcement (SCB)	0.078	**Expected** — Represents real-time policy evaluation over identity attributes. Slightly heavier than ingestion-side ABAC, due to more complex attribute matching logic in update distribution context.
Patch Query & Delivery to Device	0.090	**Expected** — Covers secure data transmission and any final policy-constrained dispatch. Timing is consistent with network-based payload delivery operations.
Total End-to-End Time	5.269	Full end-to-end duration remains in line with previous pipelines. As in other flows, Besu's on-chain write latency dominates. Overall timing is consistent with system design targets.

6.2.3.2 Observations and Insights

The performance of the threat intelligence distribution pipeline confirms the robustness, traceability, and low-latency execution of REWIRE's secure update mechanism. The end-to-end storage and dissemination of SW/FW update, anchored on-chain and distributed through the SCB, proves both efficient and scalable. As expected, the primary contributor to overall latency remains the on-chain anchoring operation within Besu, which consumes approximately 4.715 seconds of the total 5.269-second end-to-end execution time. This delay reflects the consensus finality required by the PoA-based ledger, which guarantees integrity and auditability. More importantly, the second half of the process, triggered by the SCB's WebSocket-based notification to devices, is completed in under 0.35 seconds. This includes:

- Device-initiated query for patch availability
- Signature verification of device identity via W3C-compliant Verifiable Presentations
- ABAC enforcement
- Final patch delivery

This highlights the low-latency, high-assurance interaction model enabled by the SCB, whose policy engine and identity verification mechanisms introduce minimal overhead. Together, these observations validate that REWIRE's infrastructure achieves the following key goals:

- **Cryptographic Verifiability**: All SW/FW update is securely stored with cryptographic proofs and is independently verifiable via the Blockchain.

- **Selective, Policy-Compliant Access**: Only devices that satisfy predefined ABAC policies and present verifiable identity credentials are allowed to receive updates.
- **Bounded, Predictable Latency**: Despite the cryptographic and consensus-related assurances, overall response times remain within predictable thresholds suitable for real-world deployments.

To sum up, the integration of secure enclave-based validation, policy-driven control logic, and decentralized event-driven messaging enables efficient, trustworthy threat intelligence propagation, aligning with REWIRE's vision for resilient, secure edge environments.

6.2.4 Security Policy Distribution & Enforcement - Scenario 4

The security policy distribution and enforcement workflow in REWIRE is centered around the dynamic management of Manufacturer Usage Description (MUD) profiles. These profiles act as behavioral blueprints for edge devices, specifying operational constraints and communication rules. They evolve over time to reflect changing configurations, trust levels, and threat intelligence.

6.2.4.1 MUD Profile Update Benchmark

REWIRE's infrastructure ensures that MUD profiles are stored immutably via the Besu and delivered securely via the SCB, with all operations governed by cryptographic verification and ABAC enforcement. The SCB acts as the central orchestrator, handling profile submissions, validations, and dissemination triggers. The process is composed of two tightly coupled phases:

- In the **ingestion phase**, a new or updated profile is submitted by the Policy Server to the SCB. The SCB verifies the policy using W3C-compliant Verifiable Presentations (VPs), applies attribute-based access policies, and submits the verified transaction to the Besu ledger for permanent anchoring.
- In the **retrieval phase**, devices receive a notification and initiate a query to the SCB. Their credentials are again verified through VPs, followed by ABAC policy checks, before the profile is retrieved from the ledger and securely delivered to the requester.

While the architecture can support both individualized and broad dissemination of policies, the current evaluation focuses on standard device-initiated access patterns that occur following a profile update. These support REWIRE's broader goals of ensuring verifiable, fine-grained, and dynamically adjustable policy enforcement across distributed, trust-sensitive environments. Figure 6.6 illustrates the complete end-to-end process, from policy publication to secured device delivery.

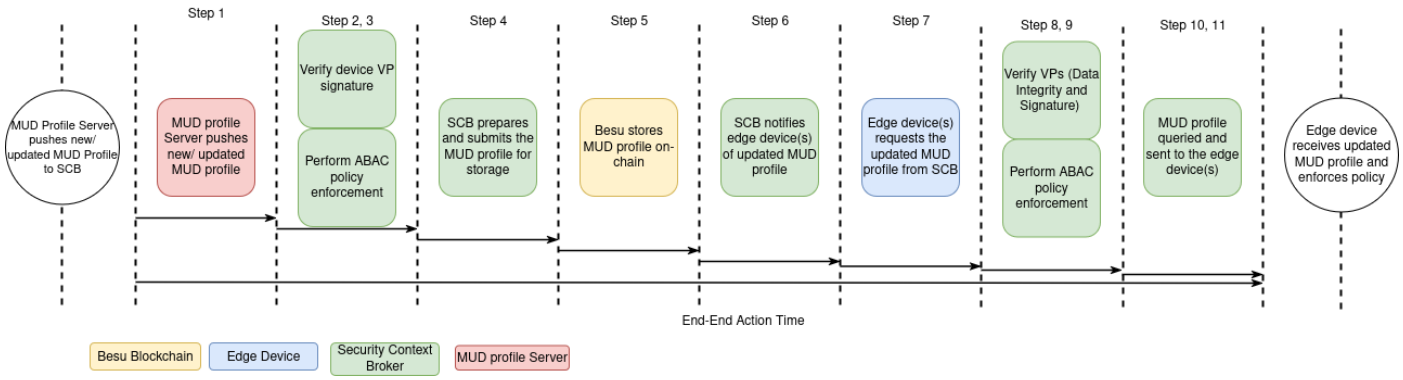

Figure 6.6: End-to-End Lifecycle of MUD Profile Update and Enforcement in REWIRE

Evaluation Scope and Step Mapping The MUD Profile Update and Distribution Flow, illustrated in Figure 6.6, reflects REWIRE's integrated process for securely updating device-level security postures based on evolving threat conditions. This dual-phase sequence (update ingestion and query distribution) follows the architectural principles defined in Chapter 2 and is decomposed into the following performance-evaluated steps:

- **Step 1: MUD Profile Server pushes new/updated MUD Profile to SCB** → This marks the initiation of the write process from the MUD Profile Server. It involves secure submission of a new or revised MUD profile to the SCB for processing.
- **Step 2: Signature Verification (SCB)** → The SCB validates the authenticity and origin of the incoming MUD profile using the corresponding Verifiable Presentation (VP).
- **Step 3: ABAC Enforcement (SCB)** → Attribute-Based Access Control (ABAC) policies are enforced to determine whether the profile can be accepted and forwarded for storage.
- **Step 4: Transaction Preparation** → The SCB transforms the verified MUD profile into a Blockchain-ready transaction structure
- **Step 5: Besu stores MUD profile on-chain** → The prepared transaction is submitted and stored into the REWIRE Blockchain using the Besu ledger.
- **Step 6: SCB notifies edge device(s) of updated MUD profile** → The SCB disseminates an update notification to relevant edge devices, triggering the start of the MUD profile query path.
- **Step 7: Initial MUD Profile Query from Edge Device to SCB** → The edge device initiates a secure HTTPS request to the SCB to retrieve the updated MUD profile.
- **Step 8: Signature Verification (SCB)** → The SCB verifies the device's identity and authorization to access the profile using the submitted VP.
- **Step 9: ABAC Enforcement (SCB)** → The SCB applies access control policies to evaluate whether the requesting device meets the access requirements for this profile.
- **Step 10: Query to Besu Ledger** → The SCB queries the on-chain storage to retrieve the most recent version of the MUD profile.
- **Step 11: Response Forwarding to Edge Device** → The SCB sends the verified and retrieved MUD profile to the requesting edge device for policy enforcement.

Measured Execution Times This evaluation sequence captures REWIRE's hybrid design pattern for secure device profiling and policy distribution, combining off-chain verification, on-chain immutability, and edge-level enforcement. The dual-path performance evaluation, covering both the policy update (ingestion) and its subsequent distribution to the devices (retrieval), reinforces the system's capacity to deliver responsive and trustworthy MUD profile propagation even under dynamic threat conditions and evolving device contexts. The performance of this MUD profile interaction flow was measured across all functional stages, from profile ingestion to device-specific querying. The steps in Table 6.11 reflect precise timings and illustrate the responsiveness of REWIRE's security policy enforcement logic.

Table 6.11: MUD Profile Update

Step	Execution Time (s)	Definition / Point of Interest

Table 6.11: MUD Profile Update

Step	Execution Time (s)	Definition / Point of Interest
Initial Profile Push from MUD Server to SCB	0.080	**Expected** — Standard HTTPS submission from the MUD authority to the SCB. Latency aligns with authenticated network API calls; no back-end computation at this point.
Signature Verification (SCB)	0.072	**Slightly Slower than TEE-based** — Signature validation using VP on the SCB side. Time is slightly higher than enclave-based equivalents, as expected due to absence of crypto acceleration.
ABAC Enforcement (SCB)	0.074	**Expected** — Policy enforcement based on identity and profile metadata. The latency reflects dynamic policy evaluation, comparable to other ABAC steps across REWIRE.
Transaction Preparation (SCB)	0.009	**Expected** — A lightweight and deterministic step for preparing data to fit the expected blockchain payload format. Minimal overhead, as anticipated.
On-Chain Storage (Besu)	4.612	**Expected** — This latency is consistent with the PoA consensus delay in Besu. Matches other anchoring flows discussed previously, where block finality dominates the performance.
Notification via SCB to Edge Device(s)	-	This is an asynchronous WebSocket-based notification. It is not part of the critical execution path and varies depending on runtime conditions, so it's excluded from precise timing.
Edge Device Initiates Query to SCB	0.070	**Expected** — Represents the HTTPS request from the edge component to SCB. Similar to other query initiations; latency is within expected range.
Signature Verification of Device (SCB)	0.038	**Expected** — Slightly faster than ingestion-side VP checks, possibly due to shorter or pre-validated device credentials. Matches known variation from enclave-free verification routines.
ABAC Enforcement for Device (SCB)	0.043	**Expected** — Identity-based access control consistent with other query-side ABAC operations. No unusual complexity, so the timing is well-aligned with design expectations.
Query to Besu Ledger	0.065	**Expected** — Read operations from Besu are fast due to indexing optimizations. This timing aligns with previous Besu query measurements for attestation or patch-related metadata.
Response Forwarding to Edge Device	0.015	**Expected** — Basic data dispatch via SCB to requester. Network I/O only, with minimal processing. Latency is stable and predictable.
Total End-to-End Time	5.078	Full end-to-end execution, from profile push to device-side retrieval, is consistent with REWIRE's trust model. Besu consensus remains the bottleneck, while other logic performs within optimal margins.

Measured Execution Times - Concurrent Query Evaluation To evaluate the system's scalability and performance under concurrent access, multiple edge devices were simulated to request the latest MUD profile in parallel. The results in Table 6.12 reflect the average execution time per query across different load levels. These findings demonstrate REWIRE's capacity to scale securely without introducing significant query delay, even under large-scale policy dissemination workloads.

Table 6.12: Concurrent Query Evaluation

Concurrent Queries (Devices)	Average Execution Time (s)
10	0.162
100	0.176
300	0.191
500	0.217
1000	0.264

6.2.4.2 Observations and Insights

The evaluation of REWIRE's MUD profile distribution and enforcement flow reveals key insights at the intersection of system design, policy control, and runtime performance. To begin with, the on-chain storage phase, although the most time-consuming due to its reliance on Blockchain consensus (~ 4.612 seconds), performs reliably and deterministically. This latency is inherent to the operation of permissioned Blockchain infrastructures like Hyperledger Besu, which must validate transactions, reach consensus, and finalize blocks. Importantly, this delay is incurred only once per MUD profile update, not per device, making it a justified trade-off in exchange for immutability, auditability, and verifiable state tracking over time. In contrast, the retrieval phase, triggered by edge devices upon receiving update notifications, is extremely lightweight. All operations, including VP signature verification, ABAC enforcement, and querying the Besu ledger, complete within ~ 0.2 seconds. This demonstrates the architectural efficiency of the SCB as a security mediator and ensures that devices can swiftly adapt their security posture in response to evolving threat contexts or policy refinements. One of the architectural strengths of this workflow lies in the clear separation of the ingestion and retrieval paths:

- The store path, initiated by the MUD Profile Server and managed by the SCB, ensures that the updated security policies are verifiably stored and anchored into the REWIRE ledger.
- The query path, triggered asynchronously by edge devices, enforces identity-aware, access-controlled retrieval of these policies, ensuring that only authorized actors access sensitive behavioral constraints.

The notification mechanism (Step 6) is especially important in maintaining this separation. While it introduces no measurable latency, it acts as a bridge between both halves of the flow, ensuring that edge devices are timely informed of updates without requiring polling mechanisms or direct Blockchain monitoring. This improves both responsiveness and resource efficiency on the device side. From a scalability perspective, REWIRE performs exceptionally well. As demonstrated in the concurrent query evaluation, the system supports up to 1,000 parallel requests with only a modest increase in average query time (~ 100 ms from baseline). This illustrates the effectiveness of SCB's internal processing logic—including signature validation, policy enforcement, and caching—and the stable performance of the Besu ledger under load. From a security and policy governance viewpoint, this workflow exemplifies REWIRE's commitment to fine-grained, decentralized, and verifiable access control. Each query is gated by VP validation and ABAC filtering, ensuring that only legitimate, cryptographically proven entities can retrieve active policy information. The use of W3C Verifiable Credentials aligns this with the broader Self-Sovereign Identity (SSI) framework promoted in REWIRE. Finally, the modular separation of storage and access logic paves the way for future extensibility. Additional enforcement mechanisms, such as compliance auditing, usage logging, or conditional access rules, can be integrated into either the SCB logic or smart contracts on Besu without architectural redesign. In summary, this workflow demonstrates a powerful combination of:

- Security assurance, through cryptographic credential validation and policy checks;
- Operational efficiency, through minimal per-device latency;

- System scalability, through parallel handling of requests;
- And architectural flexibility, through well-defined interfaces between policy ingestion and enforcement.

REWIRE thus validates that secure, policy-driven IoT management at scale is not only feasible, but practically performant and standards-compliant.

6.3 Discussion & Critique

This section reflects on the **performance**, **scalability**, **architectural coherence**, and **limitations** of the REWIRE Blockchain infrastructure based on the extensive measurements and flow evaluations conducted in the previous sections. The system, as implemented, successfully demonstrates a c**omplete and secure data life cycle**, covering ingestion, verification, storage, query, and enforcement mechanisms, all tightly bound to REWIRE's principles of trust, veracity, and decentralized policy management.

6.3.1 Performance Evaluation Summary

Across all major Blockchain-enabled data management flows and scenarios the infrastructure exhibits acceptable and predictable execution times.

- **Data ingestion flows** (e.g., application and attestation data ingestion) consistently achieve completion within $\sim$5.4s to $\sim$6s, depending on whether additional post-processing (e.g., FPC storage or device notification) is involved.
- **Query operations**, especially for recent data retrieval, are exceptionally lightweight (sub-0.2s end-to-end on average), which is crucial for real-time decision-making components.
- **Concurrent access testing** for MUD profile retrieval illustrates strong scalability: from 10 to 1000 simultaneous requests, average latency increased by only $\sim$100ms, highlighting the robustness of the SCB as a transaction broker.

The measured delays are largely attributable to Blockchain confirmation steps (Besu), which dominate overall latency. This is an expected outcome and consistent with performance in permissioned Blockchain systems using proof-of-authority (PoA)-like consensus. Importantly, most of these costly steps are tied to write operations, which are far less frequent than queries in real-world scenarios.

6.3.1.1 Consensus Mechanisms and Their Impact on Write Latency

The latency observed in the REWIRE platform's Blockchain write operations—particularly during application, attestation, and MUD profile storage, can be primarily attributed to the underlying consensus mechanism of the Blockchain infrastructure. REWIRE uses the Hyperledger Besu client, operating under a Proof-of-Authority (PoA) consensus model. This setup is consistent with many permissioned, consortium-led Blockchains where validator identities are trusted, and rapid finality is required.

Proof-of-Authority (PoA) in REWIRE PoA offers predictable block times and deterministic finality, which is crucial for trust-critical systems because it allows better monitoring of the sequence of operations. In REWIRE, Besu is configured with a block interval of approximately 5 seconds, which aligns with the 4.6–4.7 second delays observed in write-heavy operations (e.g., storing application or attestation data on-chain). This latency is not a limitation but a by-product of the security and finality guarantees offered by the PoA protocol.

- **No mining competition**: Validators are pre-authorized entities.
- **Block time determinism**: Block intervals are fixed and enforced.
- **Finality**: Transactions are typically finalized within 1–2 blocks.

The behavior is predictable and acceptable in REWIRE's context, where high integrity is prioritized over ultra-low latency for write operations.

Table 6.13: Comparison with Other Consensus Models [3]

Consensus Mechanism	Block Finality	Latency	Energy Efficiency
PoA (e.g., Besu)	Deterministic (1–2 blocks)	5s per write	High
PoS (e.g., Ethereum 2.0)	Probabilistic (1–2 min)	Variable	High
PoW (e.g., Bitcoin)	Probabilistic (6+ blocks)	Very high	Very low
PBFT (e.g., Tendermint)	Deterministic	Sub-second	High

FPC Enclave Storage and HLF Parallel The FPC subsystem—used in REWIRE to store sensitive attestation data within a trusted execution environment, leverages a deterministic consensus pattern akin to that of Hyperledger Fabric, particularly through Raft- or PBFT-style ordering services. While HLF offers sub-second block finality under typical conditions, the observed latency of approximately 0.08–0.09 seconds for FPC storage and on-chain verification is consistent with both enclave processing overhead and internal ordering logic. This performance characteristic contrasts notably with the Besu ledger's PoA consensus, where each write operation introduces a latency of approximately 4.6–4.7 seconds, reflecting the consensus-driven nature of public-chain anchoring.

Table 6.14: Ledger Comparison [2]

Ledger Type	Consensus Mechanism	Write Latency	Notes
Besu	PoA (Clique/Istanbul)	4.6–4.7s	Suitable for infrequent but auditable anchoring
FPC/HLF	Raft/PBFT-like	0.08–0.09s	Fast enclave-backed writes, ideal for confidential, frequent operations

However, while FPC offers lower per-operation latency, it introduces different performance trade-offs. Specifically, FPC's internal ordering process can become a bottleneck under high request frequency. This is due to the scheduling and endorsement logic that governs chaincode execution and transaction sequencing. As throughput demands increase, this internal ordering queue may saturate, impacting predictability and availability, especially in scenarios requiring high-frequency writes with strong reliability guarantees. In essence, while FPC excels in low-latency, confidential storage, its current scalability is bounded by the coordination overhead of its endorsement and ordering services. This makes it highly suitable for moderate-load trust evaluation pipelines, but less optimal for high-frequency, high-throughput environments without additional optimization or horizontal scaling strategies.

Conclusion The REWIRE architecture intentionally separates frequent, lightweight query operations, which bypass consensus entirely, from infrequent but security-critical write operations, which are anchored through ledger consensus. This separation allows the system to leverage the strengths of each Blockchain subsystem:

- PoA-based Besu provides auditability and tamper-evidence for control-plane events such as software updates or attestation handovers.
- FPC ensures real-time, confidential, and access-controlled storage for sensitive trust-related data.

As such, the observed performance characteristics reflect both the expected behavior of their respective consensus mechanisms and the strategic design of REWIRE's hybrid trust and policy infrastructure. While FPC's lower latency makes it optimal for secure and reactive processing, care must be taken to manage its ordering bottlenecks under high concurrency, especially in mission-critical applications with tight reliability and timing constraints.

6.3.2 Architecture-Level Observations

The REWIRE Blockchain infrastructure demonstrates a modular, secure-by-design architecture that connects distinct components (e.g, TC, Besu, and FPC) through the SCB. Such an architecture enables:

- **Isolation of concerns**: Each component is responsible for a distinct trust or compute domain (e.g., SGX-based secure data processing, ledger consensus, or confidential storage).
- **Event-driven coordination**: Storage and update flows are triggered by on-chain events observed and propagated by the TC relay, ensuring automation without centralized data ownership.
- **Integrated policy enforcement**: ABAC filtering and SSI-based VP verification are systematically applied during both consumption and querying workflows.

However, this integration strategy introduces architectural complexity. The orchestration across TC, Besu, and FPC forms a tightly coupled pipeline where each additional coordination step adds latency, synchronization overhead, and a potential point of failure, especially in distributed or resource-constrained edge environments where enclave reinitialization or network jitter may cause non-deterministic delays. Furthermore, while the SCB's centralized role is functionally justified for enforcing access policies and managing transaction orchestration, it introduces a trade-off with the decentralization principles of blockchain. Although the SCB does not modify or persist data directly, its orchestration logic becomes a critical path in workflow execution, and faults at this layer could propagate delays across the system. As previously discussed, this architecture should not be interpreted as the final convergence model but rather as a stepping stone toward the construction of a harmonized Secure Oracle Layer, in which multi-oracle environments (e.g., TC, FPC, or emerging platforms like Phala Network) interoperate more directly. In future iterations, such convergence may occur explicitly and securely, without relying on intermediate routing through a blockchain infrastructure. This vision sets the foundation for a more scalable and streamlined cross-oracle trust fabric.

6.3.3 Strengths and Capabilities

Several noteworthy strengths emerged from the evaluation:

- **Strong alignment with SSI principles**: All secure interactions are gated by VP verification and ABAC enforcement, meeting REWIRE's privacy-preserving trust goals.
- **Unified event-driven processing**: The use of Blockchain events as triggers ensures asynchronous, loosely-coupled integration between TC and SCB, enabling scalable and extensible flow definitions.
- **Confidentiality assurance via enclaves**: By leveraging both Intel SGX (Town Crier and FPC) and verified enclave measurement checks, data remains protected even in hostile or compromised environments.

- **Highly efficient querying**: Especially in the attestation and MUD profile flows, where results are returned rapidly and access control is seamlessly enforced inline.

These strengths validate that REWIRE's architecture can reliably operate across multiple data domains (application, attestation, trust policies), and handle varied interaction patterns (push, query, subscribe) while maintaining verifiability and auditability.

6.3.4 Limitations and Areas for Improvement

Despite the successful integration and performance of the REWIRE Blockchain stack, several limitations have been identified that highlight areas for future enhancement:

- **Complexity of Cross-Oracle Orchestration**: The current design depends on coordinated execution between multiple secure enclaves, TC and FPC, along with SCB mediation and Besu anchoring. However, due to current technical constraints, the envisioned direct enclave-to-enclave interaction (e.g., TC $\rightarrow$ FPC) could not be realized. Instead, attestation data flows through Besu as an intermediary. This introduces additional latency and complexity in synchronizing event propagation, particularly between trusted (enclave) and untrusted (relay or external) components.
- **Fragmentation in Enclave Execution Models**: In REWIRE, both secure oracles are based on Intel SGX, meaning the underlying TEE technology is not divergent. However, fragmentation arises from the isolation and divergence in their execution models and enclave abstractions. TC relies on a legacy SGX SDK with asynchronous fetch and processing logic, while FPC adopts Hyperledger's chaincode interface implemented in Go, aligned with Fabric's endorsement and transaction flow. This architectural fragmentation complicates the harmonization of core operations such as VP verification, secure I/O handling, and enclave-to-enclave communication, impeding the realization of a uniform, tightly integrated TCB.
- **Besu's On-Chain Query Scalability**: In data flows involving large historical datasets (e.g., timestamp-range application queries or bulk MUD profile lookups), Besu's native querying performance can degrade due to index traversal overhead. While acceptable for current load levels, scaling to denser environments will require optimized SCB-side caching, pre-filtering, or off-chain indexing strategies.
- **Trusted/Untrusted Domain Separation**: All enclave interactions involve transitions between the trusted (SGX-protected) and untrusted world (relays, SCB HTTP handlers). This creates an implicit risk surface in the data handoff points, though mitigated via encryption and cryptographic validation.

6.3.5 Strategic Design Choices and Justifications

Several architectural decisions were made intentionally to balance security, scalability, and integration complexity across REWIRE's Blockchain stack:

- **SCB as the Central Coordinator**: The SCB serves as the policy enforcement and credential validation point across all data flows. Its centralized role, though introducing a single logical coordination point, was chosen to: (a) Enforce fine-grained ABAC policies before triggering storage or query operations. (b) Reduce the attack surface by validating VPs centrally. (c) Enable context-aware processing of use-case-specific triggers (e.g., application data readiness, SW/FW update broadcasts).
- **Dual-Oracles with Complementary Roles**: REWIRE's use of both TC and FPC reflects a layered trust design: (a) TC is responsible for verifiable ingestion of data from external sources into the on-chain system. (b) FPC manages confidential evidence storage, policy-aware query handling, and real-time validation within its SGX-based chaincode. Though not yet directly interconnected, both

operate inside enclave-protected TCBs, and the current Besu-mediated interaction allows secure coordination while maintaining separation of concerns.

- **Dual-Ledger Strategy**: The decision to anchor application datagrams on Besu while persisting sensitive data on FPC allows for: (a) Auditability through event transparency and deterministic ledger proofs. (b) Confidentiality by offloading sensitive data and computation to FPC's trusted enclave. This hybrid model offers security guarantees that are difficult to achieve in either fully public or purely permissioned chains alone.
- **Query Optimization via Modal Design**: Query flows intentionally bypass Town Crier and directly access Besu or FPC depending on the type of data. For example, application data and SW/FW update queries are handled directly by SCB and Besu, avoiding TC involvement as the data already resides on-chain.

This design avoids unnecessary enclave cycles and reduces latency, while keeping TC focused on off-chain data verification.

6.3.6 Concluding Remarks

The REWIRE Blockchain infrastructure has demonstrated a **modular**, **scalable**, and **secure** architecture for managing verifiable data flows in cyber-physical and mission-critical environments. Through the combined orchestration of distributed ledgers, secure enclaves, and attribute-driven access control, REWIRE addresses key challenges in data trustworthiness, confidentiality, and selective dissemination across heterogeneous edge deployments. The evaluation results confirm that:

- Data ingestion pipelines, orchestrated through the SCB and executed within SGX-protected oracles (TC and FPC), can be completed in under 5.3 seconds. The **dominant latency** contributor is **on-chain storage** via Besu ($\sim$4.7s), reflecting the expected delay derived from the consensus mechanism.
- **Query** operations, especially those executed through FPC's enclave-based ledger, exhibit consistently **low latency** ($<$300ms) even under high concurrency levels, supporting near real-time access to trust-critical information like attestation reports.
- The **SCB** functions effectively as a central orchestration layer, enforcing W3C VPs and ABAC policies, while maintaining lightweight responsiveness ($\sim$0.09s overhead). Its placement enables tight control over identity and access without overburdening performance.
- The REWIRE design supports both one-to-many and one-to-one data dissemination modalities—as seen in SW/FW update and policy distribution scenarios, ensuring scalability across IoT topologies while retaining precise access control through ABAC.

Importantly, while the current implementation mediates TC and FPC interactions via Besu due to technological constraints, this is a transitional architecture. The envisioned end-state is direct secure enclave interoperability, reducing trust boundaries and communication latency. Limitations such as incompatible enclave SDKs (e.g., TC's legacy SGX vs. FPC's trusted chaincode model) prevented full realization during this evaluation, but the functional prototype remains compliant with REWIRE's zero-trust principles. To further strengthen the framework, future development will focus on:

- Consolidating the Secure Oracle Layer into a unified enclave abstraction or shared runtime environment.
- Supporting advanced cryptographic primitives (e.g., zk-SNARKs, threshold signatures, and selective disclosure schemes).

- Enabling privacy-preserving queries over encrypted metadata using advanced indexing and policy engines.
- Extending the SCB or smart contracts with dynamic policy enforcement logic for automated compliance and adaptation.

In summary, REWIRE's Blockchain-enabled trust layer is a robust and extensible foundation for policy-compliant, privacy-preserving, and cryptographically verifiable data sharing in decentralized edge ecosystems. By blending trusted computing with fine-grained ledger orchestration, the architecture meets the demands of dynamic trust management, secure data provenance, and confidential automation, positioning REWIRE as a mature solution for secure data handling in smart cities, smart satellites, and generally in safety-critical IoT infrastructures.

Chapter 7

AI-based Misbehaviour Detection

After describing the final version of the REWIRE Blockchain Infrastructure, particularly the adoption of Secure Oracles Layer to ensure data veracity, this chapter focuses on **enhancing data trustworthiness throughout the overall data management process**. The use of REWIRE TCB, a core building block, allows the evaluation and assessment of edge devices at a host level, based on the behaviour of the devices. On top of that, another important dimension is to assess the behaviour of the applications by the underline host device. In this context, REWIRE introduces another core building block, the AI-based Misbehaviour Detection Engine (AIMDE). As previously outlined in deliverables D5.1 and D6.1, AIMDE plays a critical role in cooperative multi-agent environments, such as those envisioned by REWIRE. These systems are essential for evaluating and guaranteeing operational assurance at the application level.

The endmost goal of REWIRE is not only to check an AI-based Misbehaviour Detection system but to depict that how such a system that focus on the detection on any inconsistencies at the application behaviour can **complement the design space of the evidence based on which a trust assessment is performed**. Trust sources provide the evidence such that any trust assessment framework can draw upon those evidence to evaluate the trustworthiness of the overall system and its internal behaviour. REWIRE provides **both the attestation results and the output of AIMDE as trust evidence** based on which a trust assessment framework can draw upon to evaluate a trust expression. The trust sources are essentially provide the evidence as the key input to form opinions about a trust proposition for a behaviour of a system.

REWIRE categorises the different types of evidence adopting a **layered approach**, starting from the **host-based evidence** that refers to attributes of the underline infrastructure elements that host a service graph chain, to **network-based evidence**, capturing the application-related data representing the system behaviour. The host-based attributes can be either static evidence such as secure boot or dynamic evidence such as attestation-related evidence (e.g., CIV). The network-based attributes refer to behaviours of the system during the operation and the data exchange of the application (e.g,. AIMDE identifies any behaviour inconsistencies).

In D6.1, REWIRE focused on evaluating AIMDE using a specific type of data within the context of the Smart Satellites UC, primarily to assess the accuracy of various plausibility checks. In this deliverable, during the second round of experiments, the focus shifts to evaluating another critical dimension of AIMDE: **how inherent correlations between data fields influence the accuracy of the trust source**. This dimension is particularly important because rich, high-quality data is not always available. Therefore, it becomes essential to assess how the accuracy of AIMDE fluctuates in scenarios where data lacks expressiveness and direct identification of point anomalies is challenging. In such cases, knowledge can still be extracted by leveraging correlations between data fields. This evaluation is applied to the Agriculture domain within the Smart Cities UC, where these constraints and the need for indirect misbehaviour detection through data correlation are relevant.

Next sections summarise what has been achieved in the evaluation of AIMDE's accuracy, based on the functional specifications fleshed out in deliverable D2.2.

7.1 Functional Specification Fulfilment

This section documents the engineering stories described in D5.1 [5] and elaborates how the development and utilisation of the AI-based Misbehaviour Detection Engine (AIMDE), a Misbehaviour Detection System (MDS), successfully meet each engineering story's objectives and requirements.

Story-I: As a Service Provider (or Security Administrator), I want to be able to build my service, exploiting a consolidated view of a domain comprising trustworthy (SW/HW) functional assets.

In the context of this functional specification, REWIRE employed AIMDE as part of its entire trust assessment architecture as an **additional trust source**, which is able to **continuously monitor** (in near-real time) the operation of application-related data deriving from SW/HW assets as it pertains to the management of the deployed application workflows. Based on its evaluation, the engine demonstrated its capability to promptly identify and flag deviations or anomalies (potential misbehaviors) within this data, providing Service Providers (or Security Administrators) with critical early warnings. These identified misbehaviors are not merely raw alerts; they are **supplemented with associated confidence scores, quantifying the certainty of the detection**, and are communicated to the Risk Assessment module as actionable '**risk indicators**'. This proactive provision of detailed information on potential vulnerabilities and integrity issues directly addresses the requirement of vulnerability awareness. As such, the **AIMDE contributes to the establishment of a holistic trust source manager**, along with the security controls and attestation capabilities provided by the REWIRE TCB, empowering service providers to make well-informed decisions regarding asset reliability and security, thus directly addressing the objective and requirements of Story-I.

Story-II: As an AIMDE I want to be able to root any (device) misbehaviour observations based on the consumption of *trustworthy* and *filtered* data.

This is the reason behind the design of a secure and auditable Blockchain architecture revolving around the integration of secure oracle capable of providing data veracity. That's the reason behind the integration of secure oracles. AIMDE consumes data from a **reliable and validated source**, the REWIRE Blockchain Infrastructure (e.g., TownCrier oracle), ensuring the **integrity** and **authenticity** of the information used for misbehavior detection; addressing the need to ingest trustful data from a reliable source; and thus directly addressing the objective and requirements of Story-II. Furthermore, the AIMDE's analytical pipelines incorporate **robust filtering mechanisms** (such as outlier removal, data smoothing, feature selection, etc.) in the preprocessing steps, which are specifically designed to eliminate any irrelevant or noisy data, enabling the engine to focus on meaningful observations of device behavior. Also, the cloud-based architecture of the AIMDE is designed with **scalability and performance** in mind, allowing it to accommodate large volumes of data from numerous devices. Finally, the AIMDE operates with a dual-phase approach (a training phase where expert feedback is utilised to establish ground truth, and a continuous, near real-time detection phase). This allows the AIMDE to **continuously learn and refine its models over time**, leveraging past observations and expert validation to significantly **improve the accuracy and effectiveness** of its misbehavior detection capabilities.

Story-III: As an AIMDE, I want to be able to have access to an extended set of ML-based heuristics so as to choose the optimal classification algorithm tailored to the plausibility checks required for the target domain.

The AIMDE is designed to be highly adaptable and effective in identifying misbehavior across diverse domains by providing to its users access to a **comprehensive repository of ML-based heuristics**, encompassing a variety of classification algorithms suitable for misbehavior detection; directly addressing the objective and requirements of Story-III. This includes statistical methods (like Elliptic Envelope and Local Outlier Factor), tree-based models (such as Decision Trees and XGBoost), and neural networks (including Keras models). This diverse selection allow users to choose the most appropriate algorithm based on the characteristics of the data and the specific plausibility check, overall empowering the AIMDE to **meticulously tailor its classification algorithms** to the specific plausibility requirements, thus optimising detection accuracy and fostering a dynamic framework against evolving threats. To support users on critically evaluating the selection of the most appropriate models/algorithms for their analysis, the AIMDE provides **robust mechanisms for evaluating the efficiency and suitability** of these different algorithms. This can involve benchmarking performance on representative datasets from the target domain, utilising relevant metrics (such as accuracy, precision, recall, and F1-score). The completeness of this design space of ML analytics has already been evaluated into the context of safety critical application domains with a variety level of the expression in the received data: In the context of the Smart Satellites UC where rich data is available both point anomalies and supervised/unsupervised learning have been evaluated. Beyond that, the AIMDE as aforementioned, is evaluated in the Smart Cities UC, when data is not expressive enough to provide knowledge. Thus we focused on the correlation of data fields.

Story-IV: As a SW Service Provider, I want to be able to react efficiently to any misbehaviour detected with the provision of the appropriate patches.

The AIMDE plays a crucial role in equipping SW Service Providers with the capability to **swiftly respond to detected misbehavior** by enabling the timely identification of issues, directly addressing the objective and requirements of Story-IV. This is achieved, considering that the AIMDE is designed for **real-time monitoring and detection of misbehavior** within software services. By continuously analysing relevant data streams, the engine can identify anomalies or deviations that indicate potential breaches or operational issues within a software. In a nutshell, the evaluation translates accuracy into the object detection ratio, that is, the proportion of events of interest identified in the ground truth scenario that are also correctly flagged by AIMDE in the consolidated view scenario. The performance depends in two dimensions (a) the object density, which is the number of devices that provide the data and is directly related to the volume of data and (b) the type of plausibility checks (considering the identified threat model) which impacts the accuracy of the AIMDE. All these dimensions had already been considered in the benchmark strategy of AIMDE in D6.1. Upon identifying an anomaly and generating the associated **risk indicators** (described through the provision of confidence scores), the AIMDE communicates this information to the SW Service Providers or Security Administrators through REWIRE's Risk Assessment Module, (via API endpoints). This ensures timely awareness and enables SW Service Provider to undertake informed decisions, regarding necessary responses such as deploying patches. Finally, since the AIMDE can **support continuous monitoring** it provides the means to **evaluate the effectiveness of the deployed patches** by observing the behavior of the software after the patch has been implemented, and looking for the persistence of the original misbehavior or the introduction of new anomalies. This feedback loop enables iterative improvements and adjustments to the deployed patches, ensuring that the responses to misbehavior are effective in the long term.

7.2 Operational Model of AI-based Misbehaviour Detection Engine

The operational model of the AIMDE within the REWIRE framework is thoroughly described in D5.1 [5] and a summary of the AIMDE's internal architecture, overall pipeline and flow of actions undertaken within the REWIRE framework is presented herein.

In what follows we summarise the internal architecture of the AIMDE for completeness purposes (see Figure 7.1). AIMDE operates through a four-layered architecture. The **Presentation Layer** (built on Vue.js) providing data scientists with an intuitive UI for the design, execution and visualisation of the analytics results. The **Backend Business Logic Layer** (built on Node.js) responsible for orchestrating the flow and concurrent processing of analytical tasks. The **Execution Level Business Logic Layer** where the data manipulation takes place and which offers diverse AI/ML algorithms (supporting both Spark for large-scale data processing and Python for advanced analytics). This execution is managed by Kubernetes, ensuring a scalable and resilient environment. The **Data Access/Storage Layer** is responsible for the ingestion of data from the REWIRE Blockchain (communicating via REST APIs) and manages data storage and efficient querying using MongoDB for structured data and Elasticsearch, facilitating future access, retrieval and querying. The overall flow involves the ingestion of data, its storage and utilisation in user-designed pipeline execution, visualisation of the corresponding results, and finally providing the analysis results including risk indicators to the Risk Assessment module through REST APIs.

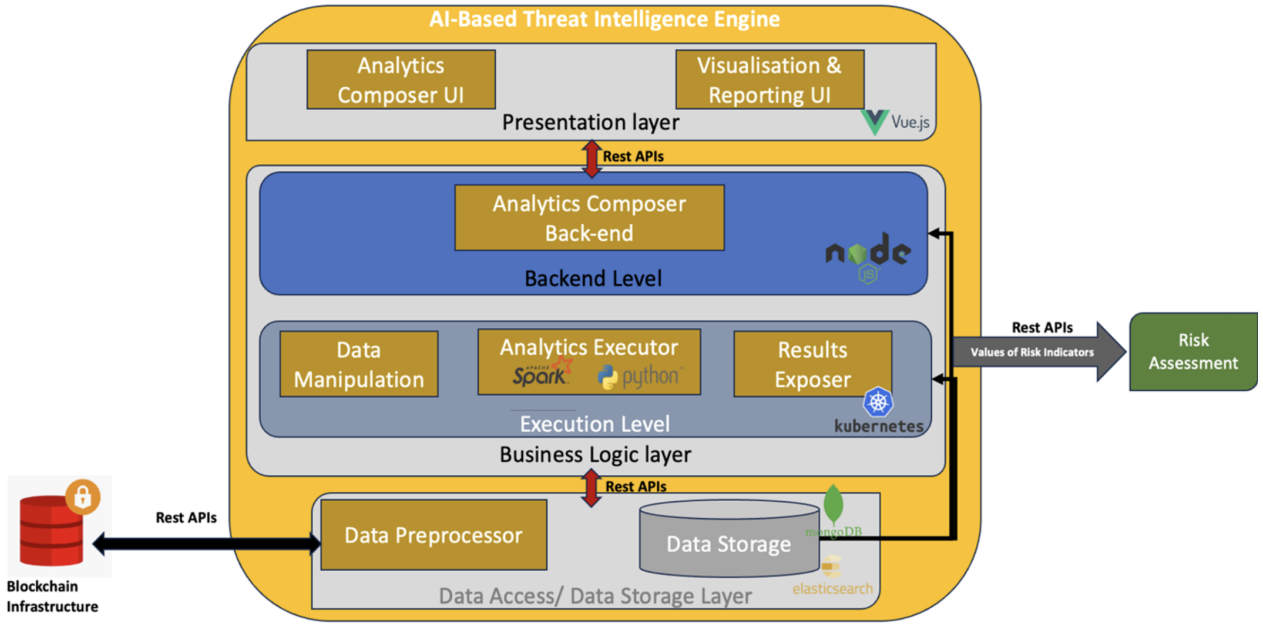

Figure 7.1: Internal architecture of the AIMDE

In respect to the flow of action for the operation of the AIMDE within the REWIRE framework; as first step the AIMDE registers and subscribes to the REWIRE Infrastructure via the SCB in order to set up a continuous watch over the relevant data streams available through the REWIRE Blockchain. This silent observation, involves also a strict access control and authentication through VCs and the attributes being checked and signed the REWIRE Blockchain Infrastructure's security components and identity management services.

Whenever a notification arrives to the engine that new application data is recorded and available, the AIMDE begins the process of gaining entry, requesting access from the SCB. Once access is obtained, the AIMDE fetches the storage pointers from the Besu blockchain and based on them, the application-related data from the Elastic Off-Chain Data Storage. The data is then decrypted (if necessary) and the engine begins its core work. This includes a detailed examination of the new data using established

plausibility checks and advanced analytical models, ultimately aiming to identify and flag any abnormal behavior (misbehavior), that might indicate a problem.

In case of identified misbehavior, the AIMDE's report is evaluated by the REWIRE Risk Assessment to analyse the potential risks associated with the detected misbehavior. More detailed can be found in Chapter 2.

7.2.1 Employed Classification Models

Within this section we present more details on the type of classification models offered by the AIMDE and which were utilised for both unsupervised and supervised learning, taking into account the plausibility checks undertaken for both the implemented Use Cases (UCs), i.e., the Smart Satellites (see D6.1 [6] for more details) and the Smart Cities UCs. The selection of the specific classification model (supervised or unsupervised) within the AIMDE is guided by factors such as the availability of labelled anomaly data (ground truth), the nature of the anomaly being targeted (point or contextual), the characteristics of the data (e.g., dimensionality, temporal dependencies), and the performance requirements for the specific UC. In this context, the Table 7.1 presents the classification models employed in both UCs along with their benefits.

ML Implemented	Benefits
Elliptic Envelope	Used in the Smart Satellites UC, to effectively identify isolated anomalies like sudden power spikes in satellite's battery capacity, crucial for detecting potential hardware or operational issues. Overall this algorithm is effective for detecting point anomalies by identifying data points that deviate significantly from the assumed elliptical distribution of the normal data.
Convolutional Neural Network (CNN) AutoEncoders	Used in the Smart Satellites UC, as these models successfully detected complex temporal anomalies indicative of satellite tumbling by learning normal patterns in sensor data (solar panels charge,vusing quaternion data), which is vital for preventing mission failures. CNN AutoEncoders are capable of learning complex temporal patterns in time series data and found effective in detecting contextual anomalies by identifying subsequences with high reconstruction errors, indicating deviations (such as satellite tumbling) from the learned "normal" behavior.
Isolation Forest (IF)	Used in the Smart Cities UC, since this algorithm offers the potential to efficiently detect both individual (point) and pattern-based (contextual) anomalies in diverse agroclimatic sensor data, aiding in the early identification of malfunctions or environmental disruptions. IF is efficient for detecting anomalies, especially in large datasets, by isolating rare instances that require fewer partitions in the isolation trees. Can be used for both point and contextual anomalies.
Local Outlier Factor (LOF)	Used i the Smart Cities UC, to identify localised deviations, such as unusual soil humidity levels compared to nearby measurements, potentially indicating sensor faults. LOF is capable of identifying anomalies based on the local density of data points. Effective for finding outliers that are significantly different from their local neighbors in terms of density.
XGBoost Classifiers	Used in the Smart Satellites UC, to effectively classify satellite passes through the South Atlantic Anomaly (SAA) based on location data, providing timely warnings. It is a supervised learning algorithm capable of learning complex relationships between features and anomaly labels and when sufficient labeled data is available for training it can achieve high accuracy in classifying anomalies. Additionally, it is used in the Smart Cities UC, to provide accurate classification of anomalies in agroclimatic data (like soil humidity and temperature) upon utilising expert-labeled data, enabling precise identification of various misbehaviors
Neural Networks	Considered for the Smart Cities UC, since these models offer the capability to learn intricate, nonlinear relationships within the agroclimatic data, potentially enabling the detection of subtle and complex anomalies that simpler methods might miss, given sufficient training data. Effective for both point and contextual anomaly detection when trained appropriately with sufficient data.
Support Vector Classifier (SVC)	SVC was successfully employed in the Smart Sattelites UC to classify satellite states related to the South Atlantic Anomaly based on latitude and longitude, demonstrating its effectiveness in distinguishing between normal and anomalous operational contexts defined by geographical location. SVC is effective for supervised classification tasks, including anomaly detection when anomalies are treated as a separate class. It aims to find an optimal boundary to separate normal and anomalous data points with a large margin.

Table 7.1: UC-Specific Benefits of AIMDE Models

Overall, the AIMDE provides access to a diverse set of models to optimise anomaly detection effectiveness across the REWIRE ecosystem. In this respect, further ML models/algorithms available though the engine include libraries like Scikit-learn (offering clustering, classification and outlier detection algorithms such as Decision Tree, KNeighbors, Bernoulli Naive Bayes, Gaussian Naive Bayes, Logistic Regression, KMeans Random Forest, XGBoost, Isolation Forest, Local Outlier Factor, Ordinal Encoderm Ine-Hot Encoder and One-Class SVM) and Keras model for neural network implementations.

7.3 Implementation Details & External Interface Specifications

The AI-based Misbehaviour Detection Engine is a component offered through the concept of a Software-as-a-Service, implemented using Python 3.10 and leverages a suite of open-source technologies, as depicted in the architectural diagram in Figure 7.1, while the libraries and technologies utilised, along with their licenses are listed in Table 7.2.

Name of the library	License
Nest NodeJS Web Framework	MIT
Elasticsearch	Elastic License 2.0
MongoDB	MongoDB, Inc.'s Server Side Public License
Elasticsearch	Elastic License
Vue.js	MIT

Table 7.2: Libraries and Technologies utilised in the AI-Based Misbehavior Detection Engine, including their Licenses

The engine interacts with other components within the REWIRE ecosystem, through external interfaces which are crucial for its operation. Specifically, a connection to the REWIRE Blockchain Infrastructure enables data queries from the distributed ledger, ensuring access to a trusted and auditable data source. In addition, an interface with the REWIRE Risk Assessment module facilitates the secure and timely provision of the analysis results (e.g., risk indicators) supplemented where possible with confidence scores. The internal interfaces of the AIMDE are thoroughly described in D5.1 [5], while Tables 7.3 and 7.4 present more detailed descriptions of these external interfaces.

Table 7.3: Data Ingestion - External interface specification

Name	Data Ingestion
Description	Ingestion of the data from the Blockchain Infrastructure (e.g., Elastic Off-Chain Data Storage)
Component/Layer	Data Preprocessor
Interface Type	REST API
Interaction with External Services	Blockchain Infrastructure (Elastic Off-Chain Data Storage)

Table 7.4: Provision of Analysis results (Risk Indicators) and Confidence scores – External interface specification

Name	Provision of analysis Results (Risk Indicators) and Confidence scores
Description	Provision of data analytics results (risk indicators) supplemented by confidence scores (where applicable)
Component/Layer	Results Exposer
Interface Type	Kafka
Interaction with Services	Risk Assessment

7.4 Performance Evaluation in Smart Cities UC

In the context of Smart Cities UC, particularly within an agroclimatic research centre in Murcia, Spain ensuring the continuous and reliable functioning of environmental monitoring and agricultural systems requires sophisticated analytical tools.

To address this, REWIRE introduced the AIMDE, capable of scrutinizing agroclimatic sensor data to identify deviations from expected environmental or operational patterns, flagging potential misbehaviors such as abnormal soil humidity fluctuations (point anomalies), unexpected shifts in indoor climate patterns (contextual anomalies), or even system malfunctions. The core focus of this evaluation lies in the engine's ability to identify and classify various types of anomalies using advanced ML/DL models.

Recall that the driving factor of this UC is to evaluate the object detection rate (accuracy) in the case where the AIMDE is fined tuned to capture trust events of interest based on any possible correlation of data. In the context of the Smart Cities UC there are no inherent structures and/or patterns that are common throughout the entire dataset. However, existing correlations within and across datasets can serve as potential candidates to enable effective plausibility checks. These checks have been specifically crafted to calculate the accuracy of the object detection ratio considering primarily these data correlations.

As described in D2.2 we examined the transfer learning from one data filed to another so that when combine and fuse data fields they will allow AIMDE to identify anomalies of the specific type. Our results, showed that the correlation of data variables that have a somewhat degree of correlation (e.g., Soil Electric Conductivity, Soil Humidity and Exterior Air Humidity) improved the object detection ratio.

Plausibility Check	Description	Indicators
Soil Humidity Fluctuations	Abnormal soil humidity readings indicating potential hardware malfunction or environmental influences. Such uncontrolled or illegitimate soil humidity fluctuations may affect the quality of agricultural production.	Sudden spike or decline in soil humidity readings, or stationary measurements unchanged for prolonged periods.

Table 7.5: Plausibility check for REWIRE AI-based Misbehaviour Detection Engine (AIMDE) in Smart Cities Use Case

The subsequent sections outline the misbehavior/plausibility check predefined for the Smart Cities UC in D2.2 [7], the experimental setup and dataset description, the phased approach undertaken to experiment with different types of anomalies, and the anticipated outcomes of these evaluations. Table 7.5, summarizes the **plausibility check** defined for this use case, pertaining to the **fluctuations in soil humidity**.

7.4.1 Experimental Setup & Dataset Description

The Smart Cities UC dataset used in the analytics process includes various sensor measurements shown in the following table, along with their corresponding data types.

7.4.2 Unsupervised Point Anomaly Detection

Initially in the data morphic and filtering process the key features where isolated, data fields with missing data were filtered out and when needed data replaced with synthetic data flowing the same distribution as the other attribute values already existing in the data profiles. Based on an extensive initial analysis of the sensor data provided by the Smart Cities UC owner (Odins), and guided by their domain expertise, the strongest **correlations were observed** between **Soil Electric Conductivity** and **Soil Humidity measurements**. As a result, the point anomaly detection focused primarily on these sensors. This

Timestamp	Date Time	Timestamp	Date Time
Relative Humidity	Float	Exterior Air Humidity	Float
Temperature	Float	Air Temperature	Float
Diffuse Solar Radiation 1	Integer	Soil Electric Conductivity 1	Integer
Diffuse Solar Radiation 2	Integer	Soil Electric Conductivity 2	Integer
Diffuse Solar Radiation 3	Integer	Soil Electric Conductivity 3	Integer
Diffuse Solar Radiation 4	Integer	Soil Electric Conductivity 4	Integer
Direct Solar Radiation 1	Float	Soil Humidity 1	Float
Direct Solar Radiation 2	Float	Soil Humidity 2	Float
Direct Solar Radiation 3	Float	Soil Humidity 3	Float
Direct Solar Radiation 4	Float	Soil Humidity 4	Float

Table 7.6: Smart Cities UC Data Inputs & data types

foundational insight directed our initial point anomaly detection efforts to these specific sensors, whose interrelationship is depicted in Figure 7.2.

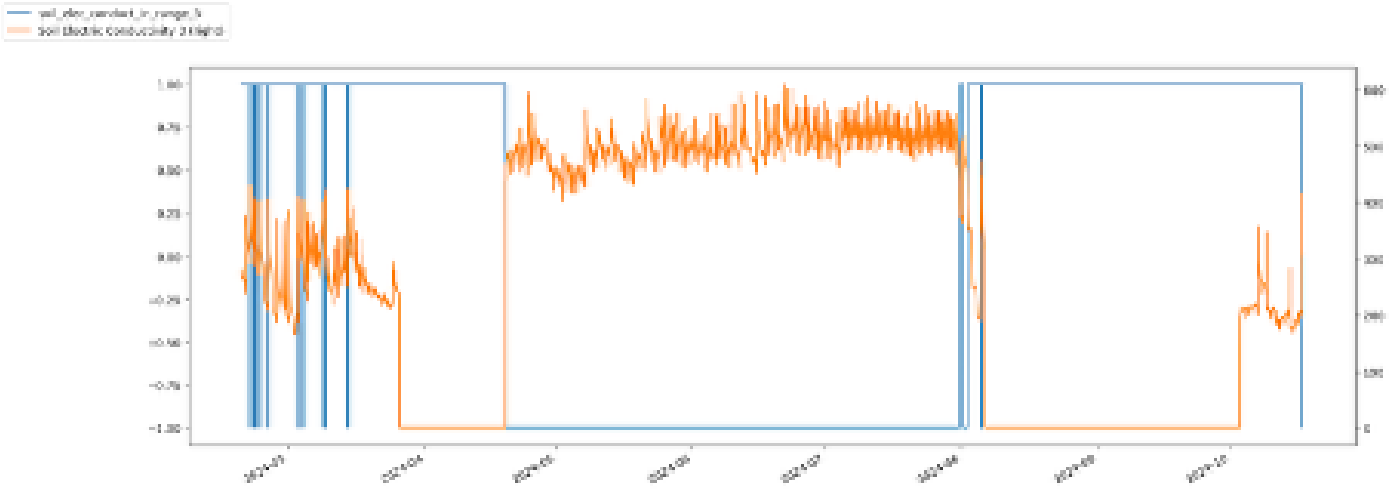

Figure 7.2: Correlation of Soil Electric Conductivity and Soil Humidity measurements

PREPROCESSING Initially, the preprocessing step focused on **selecting key features** including soil humidity, soil electrical conductivity, temperature, and air humidity. The data then underwent a **cleaning process** where extreme values, potentially detrimental to model training, were identified and clipped to fall within the 0.9 percentile. Subsequently, large segments of missing data that could not be accounted for were also removed to maintain data integrity. Finally, feature correlations were computed using the Pearson correlation coefficient [8] to assess linear interdependencies among features. A refined subset of features was selected by visually inspecting the correlation matrix and eliminating attributes that exhibited high correlations (typically above 0.9 or below –0.9) with one another, or were uncorrelated (correlations between -0.2 and 0.2). While no fixed threshold was strictly set, this qualitative approach ensured the removal of highly redundant and no informative features for the specific dataset and task, optimising the dataset for subsequent analysis. The intuition behind this process is to prevent overfitting caused by highly redundant data, or to avoid contaminating the dataset with features that lack meaningful correlation. The complete analytics pipeline for the unsupervised point anomaly detection designed in the AIMDE is shown in Figure 7.3 and the overall flow is described as follows.

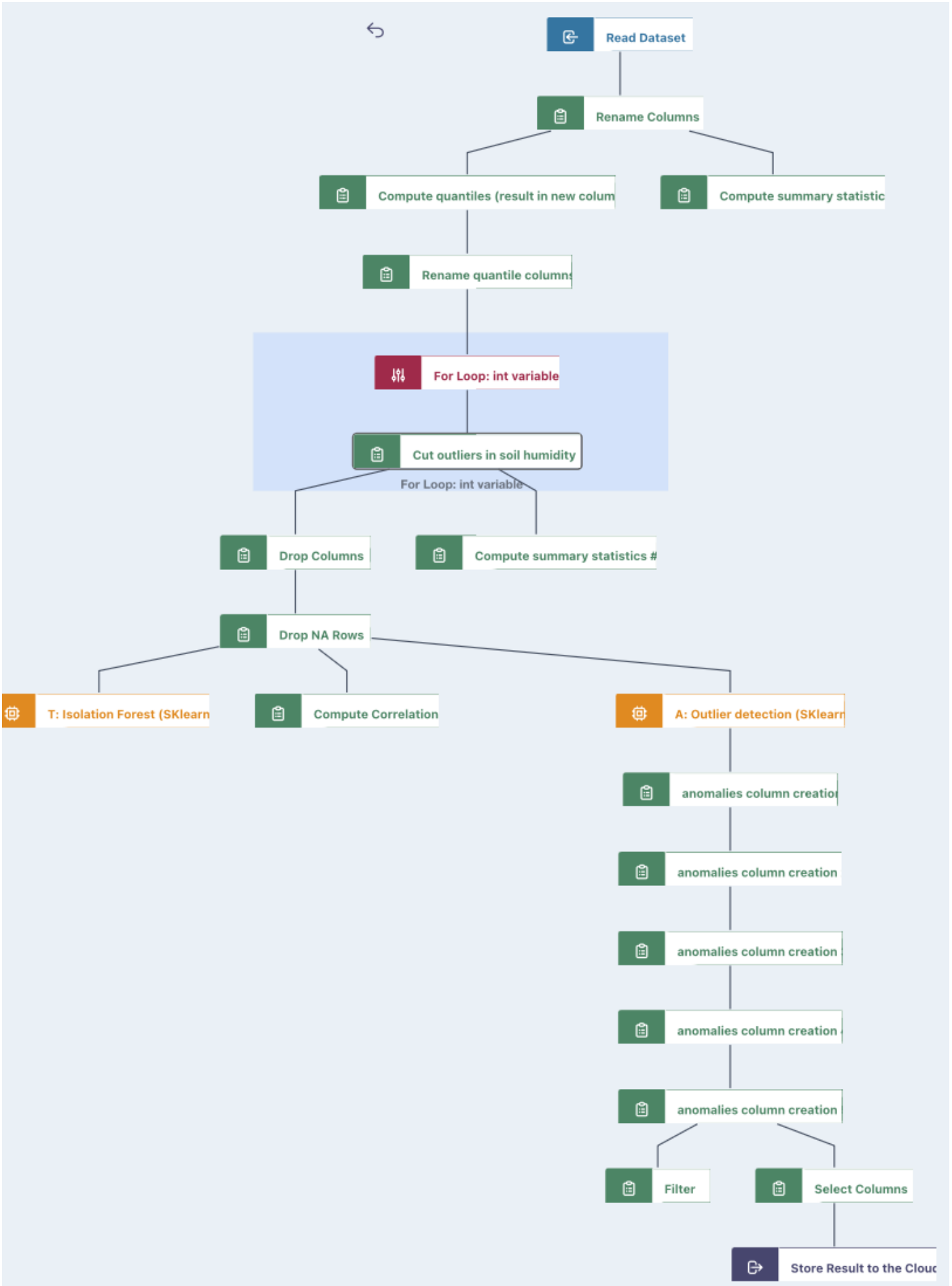

Figure 7.3: Analytics Pipeline for Unsupervised Point Anomaly Detection

More specifically, the process begins by loading and reading the agroclimatic sensor dataset into AIMDE, followed by several preprocessing steps. Initially, the dataset's columns are renamed to more meaningful and standardized names. Two parallel operations are then carried out: first, summary statistics are calculated to provide an initial overview of the dataset, including missing values and distribution characteristics. Next, quantiles are computed for selected variables to support outlier handling, and the resulting columns are renamed accordingly.

A loop is then applied to the soil humidity variable to detect and remove outliers based on the computed quantiles. Afterward, updated summary statistics are recomputed on the cleaned dataset to verify that extreme values have been appropriately addressed. Following this, unnecessary columns are dropped, and any rows containing missing values (NAs) are removed to maintain data integrity.

The workflow then branches into two analytical paths. In the first path, a Pearson correlation matrix is computed to identify relationships between variables as it can be seen in Figure 7.4. An Isolation Forest model (from Scikit-learn) is then trained to detect anomalies that disrupt these expected correlations, using features selected based on the correlation matrix. In the second path, the trained outlier detection model is applied at the individual variable level to flag anomalies. Additional steps are performed to generate a new column that labels anomalous sensor readings, and the final columns of interest (along with the anomaly labels) are selected.

#	Column	Exterior_Air_Humidity	Air_Temperature	Soil_Electric_Conductivity_1	Soil_Humidity_1	Soil_Electric_Conductivity_2	Soil_Humidity_2	Soil_Electric_Conductivity_3	Soil_Humidity_3
1	Exterior_Air_Humidity	1	-0.5959	0.3068	0.1036	0.1472	0.1368	0.1947	0.1742
2	Air_Temperature	-0.5959	1	0.0334	-0.0467	0.0027	-0.1145	0.314	0.0008
3	Soil_Electric_Conductivity_1	0.3068	0.0334	1	0.1961	0.9495	0.172	0.5877	0.2786
4	Soil_Humidity_1	0.1036	-0.0467	0.1961	1	0.1828	0.4961	0.1636	0.4971
5	Soil_Electric_Conductivity_2	0.1472	0.0027	0.9495	0.1828	1	0.1733	0.485	0.2713
6	Soil_Humidity_2	0.1368	-0.1145	0.172	0.4961	0.1733	1	0.1203	0.4826
7	Soil_Electric_Conductivity_3	0.1947	0.314	0.5877	0.1636	0.485	0.1203	1	0.3091
8	Soil_Humidity_3	0.1742	0.0008	0.2786	0.4971	0.2713	0.4826	0.3091	1

Figure 7.4: Pearson Correlation matrix among features

Finally, the cleaned and enriched dataset, including the detected anomalies, is stored in cloud storage for further use, such as visualization and forwarding to the Risk Assessment module.

MODEL TRAINING For **unsupervised point anomaly detection**, the Isolation Forest (IF) model was selected following a thorough initial investigation and experimentation with various contamination levels, i.e. the expected proportion of anomalies in the dataset (typically between 0 and 0.5), that influences how the model distinguishes between normal and anomalous data points. Given that traditional numerical evaluation metrics are not directly applicable to unsupervised learning, we relied on a combination of **domain experts' input** and rigorous **visual inspection** to assess the model's performance and its results. Essentially this whole process was culminated into the identification of values representing a realistic ground truth with the inherent deviations which was ratified together with domain experts, thus allowing for the latter evaluation of any false negative/false positive, basically the object detection ratio (the envisioned KPI).

EVALUATION By visually inspecting the results presented in Figures 7.5 and 7.6 (which depict the readings of the soil humidity sensors over time alongside the detected anomalies highlighted in light blue and identified within the red circles), it becomes clear that the Isolation Forest model, trained using the selected correlated features (soil humidity 3 and 4 and soil conductivity 3 and 4), effectively identified most of the anomalous points. The alignment of these detected anomalies with known, expert-verified deviations further supports the model's capability in capturing significant irregularities in the sensors data.

Figure 7.5: Soil Humidity Predicted Anomalies

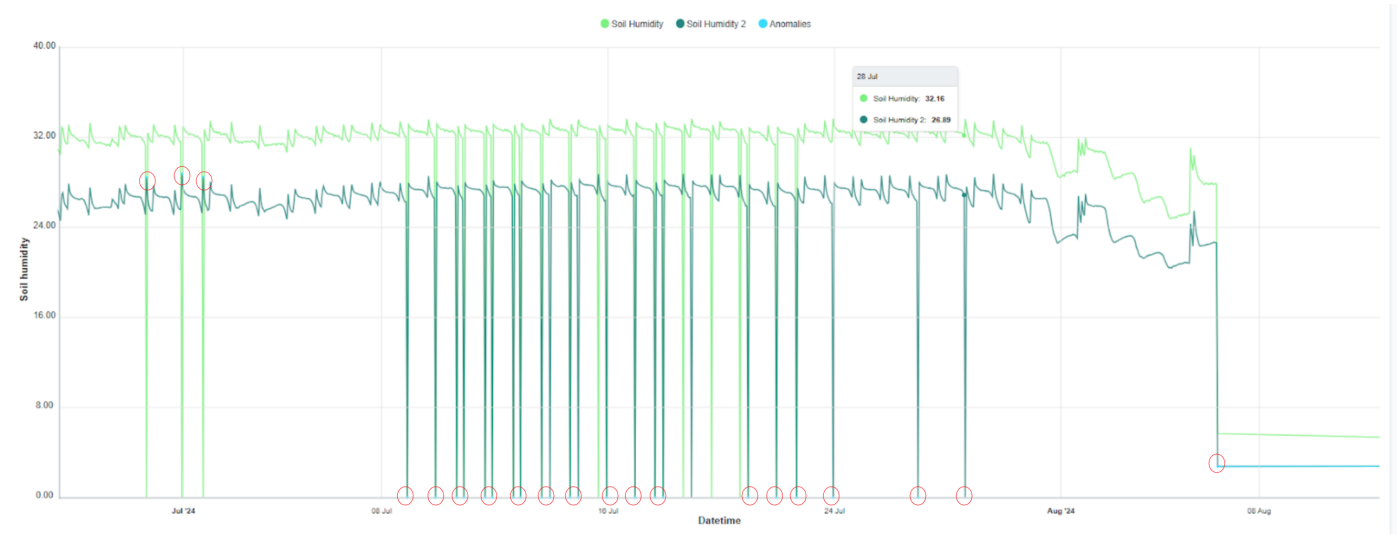

Figure 7.6: Zoomed-in View of Predicted Soil Anomalies

7.4.3 Supervised Point Anomaly Detection

After evaluating the results of the unsupervised anomaly detection approach with the expert's input, the analysis progressed with **training supervised models** for point anomaly detection, using the **previously identified anomalies as ground truth labels**. A supervised model trained on labelled data can offer improved **accuracy** and **reliability** when applied to new, unseen data, **enhancing the ability to detect** potential irregularities and misbehaviours.

PREPROCESSING Given the significant class imbalance in the original dataset ($\tilde{4}$,000 normal points vs. 139 anomalies), we applied a random sampling strategy to create a more **balanced dataset** suitable for model training. Specifically, we randomly selected 200 normal data points and combined them with the complete set of 139 anomalies, resulting in a dataset of 339 instances. This dataset was subsequently **divided into training and test sets**, comprising 229 and 110 rows respectively. **Feature selection** was informed by correlation analysis, leading to the inclusion of the input variables: **Soil Humidity 3**, **Soil Electrical Conductivity 3** and **Soil Electrical Conductivity 4**. The complete analytics pipeline designed in the AIMDE is shown in Figure 7.7.

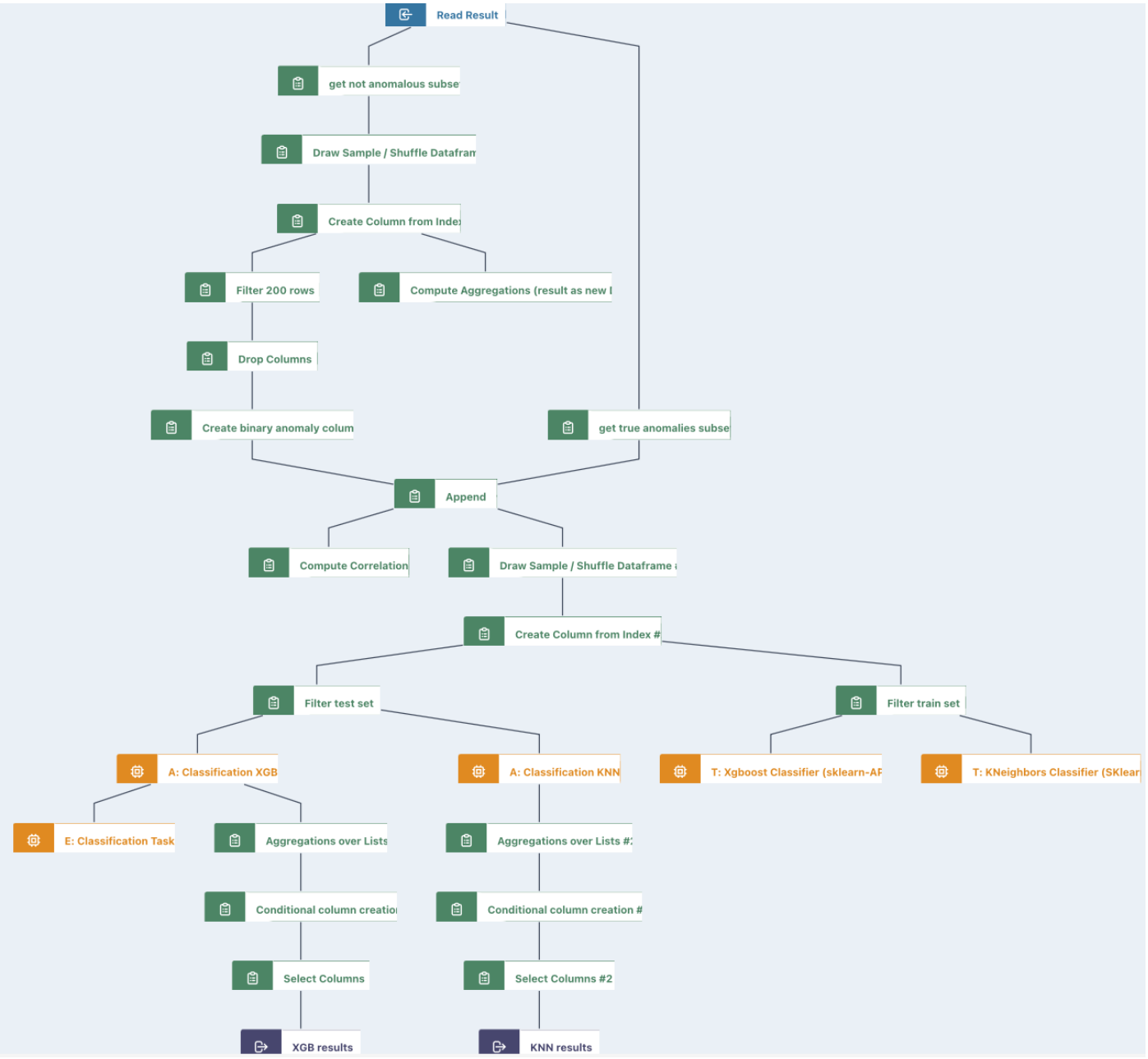

Figure 7.7: Analytics Pipeline for Supervised Point Anomaly Detection

MODEL TRAINING Subsequently, we **trained** an XGBoost classifier using the default hyperparameters and the previously selected features: Soil Humidity 3, Soil Electrical Conductivity 3, and Soil Electrical Conductivity 4.

EVALUATION The evaluation metrics presented in Figure 7.8 offer a strong indication of the effectiveness of the AIMDE in detecting the misbehavior events. In terms of **accuracy**, the AIMDE correctly identified approximately **96.4%** of all instances, including both normal and misbehaving cases. This high accuracy suggests that the model reliably distinguishes between normal and anomalous behavior in most situations. The **F1 score** (binary average), which balances both precision and recall, is **95.1%**; demonstrating that the AIMDE not only detects misbehavior accurately but also maintains a **strong balance between minimising false positives and capturing true threats or malfunctions**. This is especially important in real-world IoT environments where class imbalance or noisy signals may be present. A perfect **precision** (binary average) score of **100%** indicates that all instances flagged by the AIMDE as misbehavior was indeed a true positive. In other words, there were no false alarms, which is critical in

systems where false positives could lead to unnecessary interventions, downtime, etc. Finally, looking at the **recall** (binary average) suggests that the AIMDE successfully detected around **90.7%** of actual misbehavior cases, which means that it missed a small fraction of true positives. While this indicates there is still some room for improvement in sensitivity, the engine still **captures the vast majority of relevant threats/faults**.

#	accuracy	f1_average_binary	precision_average_binary	recall_average_binary
1	0.963963963963964	0.9512195121951219	1	0.9069767441860465

Figure 7.8: Supervised Point Anomaly detection: Evaluation metrics

7.4.4 Supervised Contextual Point Anomaly Detection

The latest step was to evaluate the impact on the accuracy of the event identification when we enrich the knowledge on the context. Thus, we enriched the attribute values with more contextual information for the humidity sensor. More specifically, considering that **humidity** sensor readings can vary depending on the time of day and the season, we explored the **impact** of incorporating **hour of the day** and **month of the year** as **additional** input **features**. These temporal variables were included in the training of the supervised anomaly detection model to evaluate their influence on detection performance.

PREPROCESSING As in the previous experiment, we constructed a balanced dataset. Additionally, we derived two time-based features: hour and **month**, from the original datetime field. Correlation analysis revealed no significant relationship between **hour** and **soil humidity**, while **month** showed a slightly higher, albeit still weak, correlation. Despite the low correlation, we included both temporal features and trained the model to evaluate its performance in practice.

MODEL TRAINING We **trained** an XGBoost classifier using default hyperparameters, incorporating **hour**, **month**, and **Soil Humidity** as input features. The complete analytics pipeline designed in the AIMDE is shown in Figure 7.9.

EVALUATION As shown Figure 7.10, although the correlation between the temporal features and humidity was weak, the classification results were nonetheless satisfactory, indicating that these features may still contribute useful context for anomaly detection. More specifically, in term of **accuracy** the AIMDE correctly classified approximately **90.1%** of all instances, indicating solid overall performance. The **F1 score** (binary average) indicates a strong balance between precision and recall at roughly **87.6%**; suggesting that the engine performed well even when needing to trade off between identifying true positives and minimising false positives. Also in terms of **precision** (binary average) the AIMDE correctly identified about 84.8% of its predicted misbehavior cases. While not perfect, this still indicates relatively low false positives, and finally the AIMDE maintained a high **recall** (average binary) of **90.7%**, successfully detecting the vast majority of actual misbehavior instances.

accuracy	f1_average_binary	precision_average_binary	recall_average_binary
0.9009009009009009	0.8764044943820225	0.8478260869565217	0.9069767441860465

Figure 7.10: Supervised Contextual Point Anomaly Detection: Evaluation Metrics

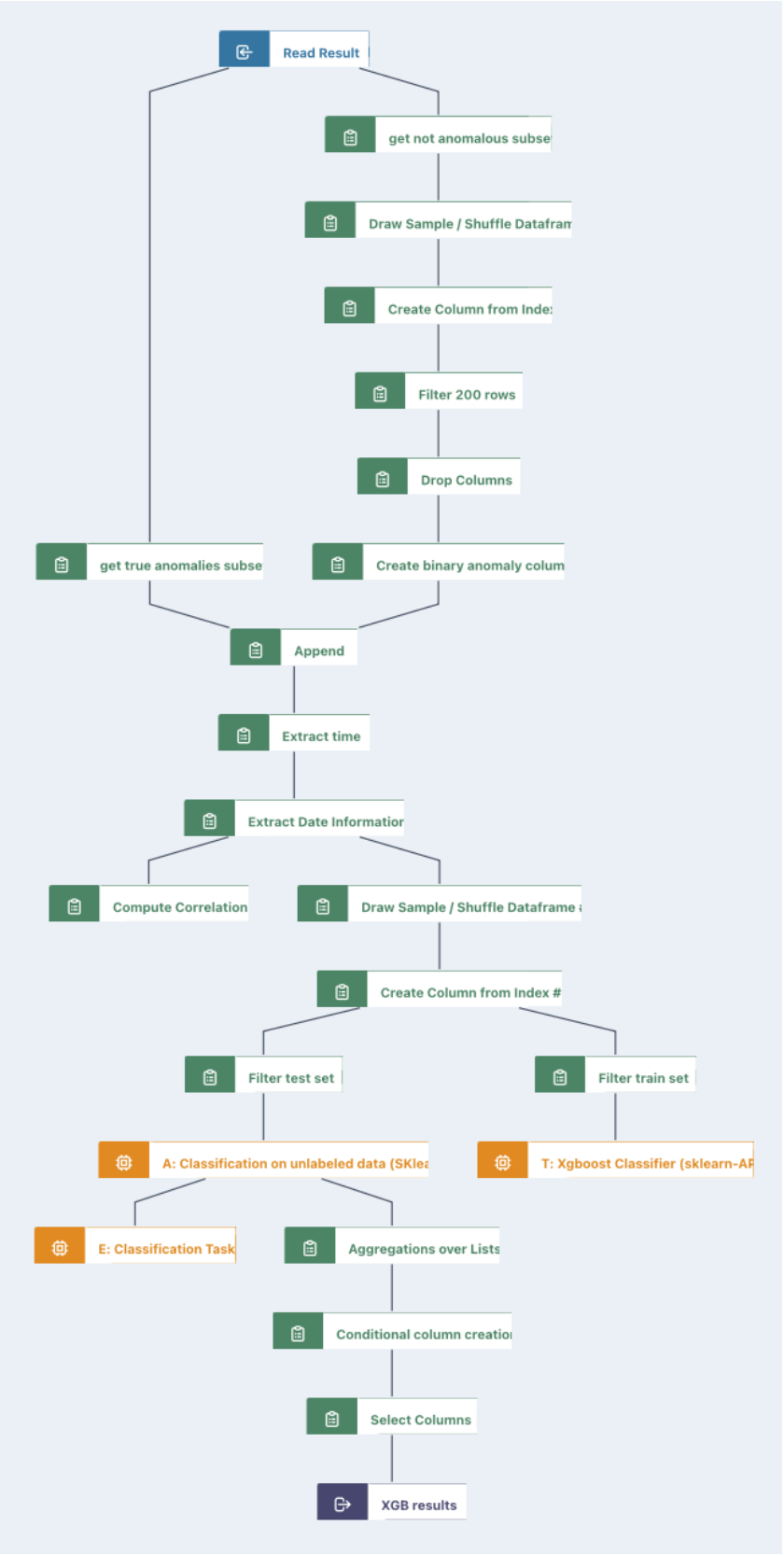

Figure 7.9: Analytics Pipeline for Supervised Contextual Anomaly Detection

7.4.5 Discussion and Critic

AIMDE demonstrates strong and **compelling evidence as a misbehaviour detection mechanism and additional trust source within a trust assessment framework**; however, its validation as a reliable trust source depends on several key dimensions. One critical dimension is the **object detection ratio** (i.e., ac-

curacy), which is highly dependent on the integrity of the plausibility checks, specifically, the correctness of the misbehaviour detectors embedded within the AIMDE pipeline. In the context of REWIRE, these plausibility checks were fine-tuned in close collaboration with domain experts, both in the Smart Satellites and Smart Cities use cases. On one hand, this collaboration highlights that access to well-designed misbehaviour detectors can significantly improve detection accuracy. On the other hand, it underscores the importance of incorporating domain-specific ground truth knowledge to ensure the effectiveness and reliability of the system.

Another dimension, as with any AI-based misbehaviour detection system, is the **lack of labelled ground truth data**, which is essential for distinguishing between normal and abnormal system behavior. Despite this limitation, our results show that AIMDE proved effective by autonomously detecting a variety of anomalous patterns in the sensor data streams, showcasing its unsupervised and supervised anomaly detection capabilities as long as there are either inherent structures and patterns within the data, or transferable correlations between data fields. In particular, we emphasize the latter capability of AIMDE to go beyond traditional point-wise anomaly detection and instead leverage correlations between data fields to enhance detection accuracy. This is critically important in smart city environments, where numerous sensors operate in tandem and influence each other. For instance, changes in soil humidity and soil electric conductivity are often interrelated. The AIMDE managed to identify this interdependency (verified also by experts' knowledge) and finally leveraged this knowledge by modelling and learning the expected relationships between sensors. When this relationship deviates significantly from established patterns, it signals a potential misbehavior, not just at the level of individual sensors, but potentially across the entire network. This "**multi-agent correlation analysis**" enables the engine to identify systemic anomalies from disruptions in the expected co-behavior of sensors rather than from isolated faults. This holistic perspective ensures a more robust detection mechanism, capable of flagging subtle and complex forms of misbehavior that rule-based or single-sensor methods might miss. The key takeaway from our evaluation of **data correlations** is the importance of correctly identifying the temporal dimension. This means that based on the degree of correlation between the data fields, a parameter that controls what is the influence of a specific data correlation on the overall misbehaviour detection must be identified. A core challenge lies in evaluating the system's appropriate response to such temporal correlations. For instance, if a correlation is weak, it should be assigned a lower weight in the detection process. Our evaluations led to the hypothesis that defining such correlation weights, based on both strength and persistence, is crucial for enhancing AIMDE's accuracy and reliability.

Finally, our results highlight the interplay and benefits of incorporating multiple sources of trust evidence in a trust assessment framework. We validated the hypothesis that leveraging multiple types of evidence can significantly improve the accuracy of such framework. AIMDE, by considering all the aforementioned dimensions, especially the fine-tuning of misbehaviour detectors and plausibility checks, provide an additional inertia in the trust assessment process (in the form of a confidence level capturing the level of certainty/uncertainty in a probabilistic trust assessment framework), enabling it to go beyond what is achievable through attestation alone. This added capability enhances the evaluation of a system's assurance level, enabling coverage of a wider threat model. As a result, AIMDE does not merely detect technical failures; it **enhances the semantic understanding of trustworthiness** in sensor-generated data by validating the consistency and integrity of the information being collected. In doing so, it offers independently **verifiable insights** that **complement other trust signals within REWIRE**. This is particularly valuable for ensuring the reliability and security of Smart Cities infrastructure, where decision-making often depends on timely and accurate sensor data, yet explicit anomaly labels are often missing or difficult to generate. However, this also manifest one of its core pending challenge on how to quantify this level of uncertainty in a systematic modelling of a trust assessment methodology capable of coping with this type of trust sources. In this context, this uncertainty primarily arises from two sources (a) from the luck of evidence or (b) the relevance/irrelevance of the trust sources that are less suitable.

Chapter 8

Conclusions

In the context of deliverable D5.2, we provided the final REWIRE Blockchain Architecture and all the Smart Contract implementation details. More specifically, the deliverable documents all the identified workflows in detail i.e. (a) Storing and Indexing Application-related Data, (b) Storing and Indexing Trust-related Evidence, (c) Application-related Data Query for Generating Threat Intelligence, (d) Attestation-related Data Query for Generating Threat Intelligence, (e) Security Policy Distribution & Enforcement and (f) MUD Profile Flow. On top of that documents all the Blockchain-dependent entities (e.g., Threat Intelligence components, SW/FW Distribution Service, MUD Profile Server etc.). Additionally, it documents all blockchain-integrated entities such as the Threat Intelligence components, SW/FW Distribution Service, and the MUD Profile Server.

Then the document focuses on the Secure Oracle Layer in REWIRE, built on three main technologies i.e. TownCrier, Hyperledger Besu, and Fabric Private Chaincode, that ensure secure and private data interactions. While effective initially, the system faces issues like limited interoperability, inflexible configurations, and fragmented execution. To address these, REWIRE is exploring unified solutions like the Phala Network, aiming to improve security, confidentiality, and flexibility. D5.2 details the various smart contract types and functions deployed on Besu, FPC, and TC. These contracts form the core of REWIRE's automated processes, ensuring secure data handling, device management, and policy enforcement in IoT ecosystems. On top of that, the deliverable focuses on the SCB, that as the central coordination and execution layer in REWIRE Blockchain Infrastructure manages secure interactions between internal components and external modules by enforcing ABAC, validating identities, and orchestrating complex transactions. In other words, SCB is acting as the main entry point for blockchain-related operations and routes external requests to backend systems, ensuring secure, structured, and policy-driven data sharing across the REWIRE ecosystem.

The primary objective of this final release of the REWIRE Blockchain is the comprehensive benchmarking of end-to-end data flows by focusing on the end-to-end performance analysis of application-level data pipeline, attestation processes, SW/fW update dissemination, and MUD profile exchange. It further incorporates micro-benchmarking of the constituent internal processing stages. Extensive stress testing and scalability assessments have been conducted, substantiating the performance efficiency and scalability characteristics of the REWIRE Blockchain software stack under high-load and large-scale deployment scenarios. In general, latency in write operation is largely influenced by the PoA consensus mechanism, while operations remain efficient with minimal delay.

Last but not least, the deliverable concludes with the evaluation results of the AI-based Misbehaviour Detection Engine (AIMDE). As thoroughly discussed in the deliverable the AIMDE is a threat intelligence component that can be used as an additional trust source in a trust assessment mechanism towards a more consolidated view on the trustworthiness of the IoT devices. REWIRE's AIMDE effectively analyzes agroclimatic sensor data to detect various types of anomalies, ranging from point anomalies (e.g., soil humidity spikes) to contextual anomalies (e.g., unexpected indoor climate changes). The engine leverages advanced ML/DL models across supervised and unsupervised settings. The conducted experiments,

in the context of the Smart Cities UC, prove that AIMDE successfully detects the anomalies with high accuracy, excellent precision, and strong recall across the multiple detection scenarios. While there is room for improved sensitivity in certain contexts, its robustness and reliability make it highly effective as an additional trust source mechanism.

Annex

8.1 Annex I - MUD Profile

Listing 8.1: Example of MUD Profile


```
{
    "muddata": {
        "ietf-mud:mud": {
            "mud-version": 1,
            "mud-url": "https://ariamanufacturer.com/device_uuid",
            "last-update": "2024-02-16T11:35:25+01:00",
            "cache-validity": 48,
            "is-supported": true,
            "modelname": "Aria",
            "systeminfo": "The Aria device Current MUD File",
            "extensions": [
                "so-hash",
                "integrity-fingerprint",
                "mud-signature-certification-authority",
                "rewire-mspl-list:mspls"
            ],
            "so-hash": "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a
                08",
            "integrity-fingerprint": "{'moore_machine': {'initial-state': '0', 'input-
                alphabet': ['0', '3', '2', '1'], 'output-alphabet': ['0', '2', '1'], '
                output-function': {'0': '0', '1': '0', '10': '2', '11': '2', '12': '0'
                , '13': '2', '14': '2', '15': '2', '16': '2', '17': '2', '18': '2', '1
                9': '2', '2': '0', '20': '2', '21': '0', '22': '0', '23': '0', '24': '
                0', '25': '0', '26': '0', '3': '0', '4': '0', '5': '2', '6': '2', '7':
                 '2', '8': '2', '9': '1'}, 'states': ['0', '26', '17', '15', '24', '19
                ', '6', '10', '1', '8', '13', '21', '12', '14', '23', '3', '5', '4', '
                22', '20', '7', '2', '11', '16', '25', '18', '9'], 'transition-
                function': {'0': {'0': '9', '1': '1', '2': '9', '3': '9'}, '1': {'0':
                '9', '1': '2', '2': '24', '3': '9'}, '10': {'0': '9', '1': '9', '2': '
                11', '3': '13'}, '11': {'0': '9', '1': '9', '2': '12', '3': '9'}, '12'
                : {'0': '9', '1': '9', '2': '9', '3': '9'}, '13': {'0': '14', '1': '9'
                , '2': '9', '3': '9'}, '14': {'0': '9', '1': '9', '2': '11', '3': '15'
                }, '15': {'0': '16', '1': '9', '2': '9', '3': '9'}, '16': {'0': '9', '
                1': '9', '2': '11', '3': '17'}, '17': {'0': '18', '1': '9', '2': '9',
                '3': '9'}, '18': {'0': '9', '1': '9', '2': '11', '3': '19'}, '19': {'0
                ': '20', '1': '9', '2': '9', '3': '9'}, '2': {'0': '10', '1': '3', '2'
                : '21', '3': '9'}, '20': {'0': '9', '1': '9', '2': '11', '3': '9'}, '2
                1': {'0': '9', '1': '22', '2': '10', '3': '9'}, '22': {'0': '9', '1':
```

```
            '23', '2': '9', '3': '9'}, '23': {'0': '14', '1': '9', '2': '9', '3':
            '9'}, '24': {'0': '9', '1': '25', '2': '9', '3': '9'}, '25': {'0': '9'
            , '1': '22', '2': '26', '3': '9'}, '26': {'0': '9', '1': '22', '2': '9
            ', '3': '9'}, '3': {'0': '9', '1': '4', '2': '9', '3': '9'}, '4': {'0'
            : '10', '1': '5', '2': '9', '3': '9'}, '5': {'0': '5', '1': '5', '2':
            '6', '3': '7'}, '6': {'0': '8', '1': '6', '2': '7', '3': '0'}, '7': {'
            0': '7', '1': '6', '2': '0', '3': '6'}, '8': {'0': '5', '1': '5', '2':
             '8', '3': '0'}, '9': {'0': '9', '1': '9', '2': '9', '3': '9'}}}, '
            moore_alpha_maps': [{'input-map': {'fclose@plt': '2', 'fopen@plt': '1'
            , 'fread@plt': '0', 'fwrite@plt': '3'}, 'output-map': {'ACCEPT': '0',
            'OK': '2', 'REJECT': '1'}}, {'input-map': {'0': 'fread@plt', '1': '
            fopen@plt', '2': 'fclose@plt', '3': 'fwrite@plt'}, 'output-map': {'0':
             'ACCEPT', '1': 'REJECT', '2': 'OK'}}]}",
        "device-threats-list": {
            "threats-lists": {
                "threat-list": [
                    {
                        "name": "cve-125",
                        "category": "software-vulnerability",
                        "impact": "high",
                        "cve": "test.url"
                    }
                ]
            },
            "from-device-policy": {
                "access-lists": {
                    "access-list": []
                },
                "mspl-list": {
                    "mspls": [
                        {
                            "name": "mspl_6ccf1735-d981-4e30-87ad-34a43d5f72fa"
                        }
                    ]
                }
            },
            "to-device-policy": {
                "access-lists": {
                    "access-list": []
                },
                "mspl-list": {
                    "mspls": [
                        {
                            "name": "mspl_6ccf1735-d981-4e30-87ad-34a43d5f72fa"
                        }
                    ]
                }
            }
        },
        "ietf-access-control-list:acls": {
            "acl": []
```

```
            },
            "rewire-mspl-list:mspls": {
                "mspl": [
                    {
                        "name": "mspl_6ccf1735-d981-4e30-87ad-34a43d5f72fa",
                        "type": "boostrapping-mspl-type",
                        "configuration": {
                            "capability": "bootstrapping",
                            "rule-set-configuration": {
                                "name": "mspl_set_11868758-2e8f-4674-8a20-685ce351d3d1",
                                "configuration-rule": [
                                    {
                                        "name": "Rule_71405326-8629-42f5-92aa-391b4de1a8
                                            41",
                                        "isCNF": false,
                                        "external-data": {
                                            "priority": 500
                                        },
                                        "configuration-action": {
                                            "bootstrapping-action": {
                                                "bootstrapping-action-type": "
                                                    BOOTSTRAPPING",
                                                "bootstrapping-model-version": "0.1",
                                                "privacy-bootstrapping-model": false
                                            }
                                        },
                                        "configuration-condition": {
                                            "bootstrapping-configuration-condition": {}
                                        }
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        }
    }
}
```


8.2 Annex II - Attestation report

Listing 8.2: Example of attestation report


```
{
  "attestationReport": [
    {
      "appraisal": "1",
      "claim": "integrity",
      "timestamp": "2025-05-08T14:00:00Z"
```

```
    }
  ],
  "digest": "4bc32e967dcbc6fc214388292d4fabdebf42e248aa7866517f4a7fd08cc1c806",
  "signature": "3044022029627f6308f0774d52165bf0cf0039c635a3bf5d76ab0842bd8c62511668d
     5f902201fe5920366ee173f99572d09800a7fd9b20f1fec2ec78bf85b48319e67c1d8d3",
  "public_key": "-----BEGIN PUBLIC KEY-----\nMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAElxJY
     68rD/Zkh0VARYrUV8T8y6oYe8rvkDHbBm1La\nKjPFrhlK8xn6P3GmIXxvyyWmzUi47ePJmcRus/w3
     vXfqOA==\n-----END PUBLIC KEY-----\n"
}
```


8.3 Annex III - MSPL SW/FW Update

Listing 8.3: Example of a SW/FW update policy expressed in MSPL

```
<ITResource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <configuration xsi:type="RuleSetConfiguration">
        <capability xsi:type="TaskCapability">
            <Name>Software Update Service</Name>
        </capability>
        <DeviceCondition>
        <isCNF>false</isCNF>
        <DeviceIDList>
                <DeviceID>R</DeviceID>
                <DeviceID>K</DeviceID>
                <DeviceID>L</DeviceID>
                <DeviceID>M</DeviceID>
                <DeviceID>N</DeviceID>
            </DeviceIDList>
            <VerifierIDList> </VerifierIDList>
        </DeviceCondition>
        <AttestationTaskList>
            <AttestationTask>
                <AttestationTaskID>0</AttestationTaskID>
                <AttestationUniqueIdentifier>XX</AttestationUniqueIdentifier>
                <AttestationTaskName>CIV</AttestationTaskName>
                <ResToBeAttested>Binary unpacking process</ResToBeAttested>
            </AttestationTask>
                <AttestationTask>
                <AttestationTaskID>2</AttestationTaskID>
                <AttestationUniqueIdentifier>KK</AttestationUniqueIdentifier>
                <AttestationTaskName>CIV</AttestationTaskName>
                <ResToBeAttested>Device SW Stack</ResToBeAttested>
            </AttestationTask>
            <AttestationTask>
                <AttestationTaskID>3</AttestationTaskID>
                <AttestationUniqueIdentifier>YY</AttestationUniqueIdentifier>
                <AttestationTaskName>CFA</AttestationTaskName>
                <ResToBeAttested>Data Chunk Encryption</ResToBeAttested>
            </AttestationTask>
        </AttestationTaskList>
```

```
        <ConfigurationRule>
            <AttestationAction>
                <TaskActionType>EXECUTE</TaskActionType>
                    <TaskActionParameters>
                        <TaskParameters>
                        <ProcessID>XX</ProcessID>
                        <ExecutionTime>1</ExecutionTime>
                        <VerifierID>R</VerifierID>
                        </TaskParameters>
                    </TaskActionParameters>
            </AttestationAction>
        </ConfigurationRule>
        <ConfigurationRule>
            <AttestationAction>
                <TaskActionType>EXECUTE</TaskActionType>
                <TaskActionParameters>
                <TaskParameters>
                    <ProcessID>SW Update Process</ProcessID>
                    <ExecutionTime>2</ExecutionTime>
                    <VerifierID>R</VerifierID>
                </TaskParameters>
                </TaskActionParameters>
            </AttestationAction>
        </ConfigurationRule>
        <ConfigurationRule>
            <AttestationAction>
            <TaskActionType>EXECUTE</TaskActionType>
            <TaskActionParameters>
                <TaskParameters>
                    <ProcessID>KK</ProcessID>
                    <ExecutionTime>3</ExecutionTime>
                    <VerifierID>R</VerifierID>
                </TaskParameters>
                </TaskActionParameters>
            </AttestationAction>
        </ConfigurationRule>
        <ConfigurationRule>
            <AttestationAction>
            <TaskActionType>EXECUTE</TaskActionType>
                <TaskActionParameters>
                    <TaskParameters>
                        <ProcessID>YY</ProcessID>
                        <ExecutionTime>4</ExecutionTime>
                        <VerifierID>R</VerifierID>
                    </TaskParameters>
                </TaskActionParameters>
            </AttestationAction>
        </ConfigurationRule>
    </configuration>
</ITResource>
```

Bibliography

[1] Shimaa A. Abdel Hakeem, Anar A. Hady, and HyungWon Kim. 5g-v2x: standardization, architecture, use cases, network-slicing, and edge-computing. *Wirel. Netw.*, 26(8):6015–6041, November 2020.

[2] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain. In *Italian Conference on Cyber Security (06/02/18)*, January 2018.

[3] Richard Li. Comparative analysis and future directions of consensus algorithms in blockchain technology. In *Proceedings of the 2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI2024)*, Advances in Computer Science Research, pages 333–344. Atlantis Press, October 16 2024.

[4] REWIRE. Rewire operational landscape, requirements, and reference architecture - initial version. Deliverable D2.1, The REWIRE Consortium, 12 2023.

[5] REWIRE. Rewire blockchain architecture, continuous authorisation, and on/off chain data management. Deliverable D5.1, The REWIRE Consortium, 04 2024.

[6] REWIRE. Rewire integrated framework (1st release) and use case analysis. Deliverable D6.1, The REWIRE Consortium, 05 2025.

[7] REWIRE. Rewire operational landscape, requirements, and reference architecture - final version. Deliverable D2.2, The REWIRE Consortium, 24 2025.

[8] Philip Sedgwick. Pearson's correlation coefficient. *Bmj*, 345, 2012.